Department of Computer Science

BSCCS Final Year Project Report

Data-driven Characterization of Fake Content in Social Media

Student Name : **FAN Cheuk Pan**

Abstract

In recent years, social media content growing rapidly. Users and content are also growing on a large scale. Many people use social media to deliver information and discuss with others. However, not all information on social media is correct.

Therefore, this project aimed to classify fake content on social media, analyze the characteristics of the fake content, and raise awareness of fake content in the public by providing convenient ways to classify fake content.

This project focuses on detecting fake content using various text embedding modes and algorithms such as BOW, TF-IDF, Word2Vec, FastText, Random Forest, SVM, Decision Tree, Logistic Regression, Naive Bayes, RNN GRU, transformer BERT, and XLNet. The implementation of the system includes a web dashboard and a Chrome plugin for end-users to perform detection tasks, supported by backend APIs and deployed on the cloud using AWS services and MongoDB atlas for the database.

Table of Content

Table of Content 3

Table of figures 7

Table of tables 11

1. Introduction 12

    1.1. Background 12

    1.2. Problem Statement 12

    1.3. Objectives 13

    1.4. Scope 14

2. Literature Review 15

    2.1. Fake content on the internet 15

    2.2. Prior method of fake content detection 15

        2.2.1. Feature-based classification 15

        2.2.2. Knowledge-based classification 16

        2.2.3. Learning-based classification 16

    2.3. Natural Language Processing 17

    2.4. Text Classification 17

        2.4.1. Machine Learning Approach 17

        2.4.2. Deep Learning Approach 18

3. Methodology 19

    3.1. Overview 19

3.2. Requirement 19
3.2.1. Functional requirement 19
3.3. System usecase 21
3.4. System structure 23
3.5. Technical component 24
3.5.1. Web application 24
3.5.2. API server 25
3.5.3. Model server 26
3.5.4. Database 27
3.6. Model training 27
3.7. Deployment 30
3.8. User activity 31
3.8.1. Web Application 31
3.8.2. Chrome Plugin 32
4. Implementation 34
4.1. Overview 34
4.2. Backend 35
4.3. Frontend 37
4.4. Data acquisition 39
4.4.1. Data Source 39
4.4.2. Data preprocessing 51

4.5. Model training 52
4.5.1. Problem framing 52
4.5.2. Model training 53
4.5.3. Evaluation Method 56
4.6. Database 56
5. Result and evaluation 59
5.1. Web Dashboard 59
5.1.1. Home Dashboard 59
5.1.2. Detection Page 60
5.1.3. Dataset Analysis 60
5.2. Chrome Plugin 62
5.3. Cloud Deployment 63
5.4. Model Result 67
5.4.1. Bag of Words 69
5.4.2. TF-IDF 71
5.4.3. Word2Vec 74
5.4.4. FastText 78
5.4.5. Transformer 83
6. Project Timeline 85
6.1. Project Stage 86
6.1.1. Phase 1: Research and Data Preprocessing 86

6.1.2. Phase 2: Analysis and Modeling 86

6.1.3. Phase 3: Model Implementation and Testing 86

7. Conclusion 88

8. Reference 89

Appendix 95

Monthly Log 95

October 95

November 96

December 97

January 98

February 99

March 100

Table of figures

Figure 1 Use case diagram..... 21

Figure 2 System architecture diagram..... 23

Figure 3 React.js logo (*React – a JavaScript Library for Building User Interfaces*, n.d.)..... 24

Figure 4 Node.js icon (Node.js, n.d.) 25

Figure 5 Express.js logo (Express - Node.js Web Application Framework, n.d.) 25

Figure 6 FastAPI logo (FastAPI. (n.d.). https://fastapi.tiangolo.com/)ss 26

Figure 7 MongoDB icon (MongoDB, n.d.)..... 27

Figure 8 Scikit learn logo (Scikit-learn: Machine Learning in Python — Scikit-learn 1.2.1 Documentation, n.d.) 28

Figure 9 Spacy logo (spaCy · Industrial-strength Natural Language Processing in Python, n.d.) 28

Figure 10 Gensim logo (Gensim: Topic Modelling for Humans, n.d.) 28

Figure 11Figure 11 HuggingFace Transformer Logo Hugging face logo. (n.d.). https://huggingface.co/brands..... 29

Figure 12 Activity diagram- web application..... 31

Figure 13 Activity diagram - chrome plugin 32

Figure 14 System overview 34

Figure 15 Backend architecture diagram..... 35

Figure 16 Wireframe - Home Page..... 37

Figure 17 Wireframe - Verification Page..... 38

Figure 18 Wireframe - Dataset Page 39
Figure 19 FakeNewsNet data process flow chart 41
Figure 20 FakeNewsNet sample data 41
Figure 21 FakeNewsNet label distribution 42
Figure 22 FakeNewsNet dataset label count 42
Figure 23 FakeNewsNet text length 43
Figure 24 Text length distribution of two label 44
Figure 25 Time series label distribution 44
Figure 26 Word Cloud of the dataset 45
Figure 27 Dataset text annotation word cloud 46
Figure 28 Top 10 entity name 47
Figure 29 Top 10 entity name without top word 47
Figure 30 Top 10 domain name 48
Figure 31 Top 10 entity name without top word 48
Figure 32 True tag word cloud 48
Figure 33 False tag word cloud 48
Figure 34 LIAR dataset information 49
Figure 35 LIAR dataset label count 50
Figure 36 LIAR dataset label distribution 50
Figure 37 LIAR text length 51
Figure 38 spaCy preprocessing pipeline (Language Processing Pipelines, n.d.) 51

Figure 39 Data preprocessing flow..... 52

Figure 40 Confusion Matrix (Daniel, 2020)..... 56

Figure 41 ER diagram 57

Figure 42 Web dashboard screenshot Home Screen..... 59

Figure 43 Web dashboard screenshot Detection Screen..... 60

Figure 44 Web dashboard screenshot Dataset Information Screen 61

Figure 45 Web dashboard screenshot Tweet Analysis Screen..... 61

Figure 46 Chrome Plugin popup button 62

Figure 47 Chrome Plugin Detected Result..... 62

Figure 48 Chrome Plugin model selection 63

Figure 49 AWS Cloud Infrastructure..... 64

Figure 50 Amazon ECR..... 65

Figure 51 Amazon EC2 Instance..... 65

Figure 52 AWS SageMaker model 66

Figure 53 AWS S3 Bucket to store the model..... 66

Figure 54 MongoDB Atlssas 66

Figure 55 Testing score result..... 68

Figure 56 BOW model result..... 69

Figure 57 BOW + Decision Tree..... 70

Figure 58 BOW + Logistic Regression 70

Figure 59 BOW + Naive Bayesss..... 70

Figure 60 BOW + SVM..... 71

Figure 61 BOW + Random Forest..... 71

Figure 62 TF-IDF Model Result..... 71

Figure 63 TF-IDF + Naive Bayes..... 72

Figure 64 TF-IDF Radom Forest..... 72

Figure 65 TF-IDF + Decision Tree..... 73

Figure 66 TF-IDF + SVM 73

Figure 67 TF-IDF + Logistic Regression 73

Figure 68 Word2Vec Benchmark Score 74

Figure 69 Word2Vec Similarity Word 75

Figure 70 Word2Vec Model Result 75

Figure 71 Word2Vec RNN..... 76

Figure 72 Word2Vec RNN training loss..... 76

Figure 73 Word2Vec Training Result 76

Figure 74 Word2Vec + Decision Tree 77

Figure 75 Word2Vec + Logistic Regression..... 77

Figure 76 Word2Vec + Naive Bayes 77

Figure 77 Word2Vec + Random Forest 78

Figure 78 Word2Vec + SVM 78

Figure 79 FastText Model Benchmark Score..... 79

Figure 80 FastText Model Result 80

Figure 81 FastText RNN 80
Figure 82 FastText RNN Training Score 80
Figure 83 FastText Training Result 81
Figure 84 FastText + SVM 81
Figure 85 FastText + Decision Tree 81
Figure 86 FastText + Logistic Regression 82
Figure 87 FastText + Naive Bayes 82
Figure 88 FastText + Naive Bayes 82
Figure 89 XLNet training result 84
Figure 90 Project timeline - Gantt chart 87

Table of tables

Table 1 Technical component 24
Table 2 API Server endpoint 35
Table 3 Model server endpoint 36
Table 4 FakeNewsNet dataset size 42
Table 5 Project timeline 85

1. Introduction

1.1. Background

In recent years, social media has grown rapidly and become an indispensable part of our life. Social media allow people to share content on the internet easily. However, some content on social media is misinformation or disinformation, which may mislead the reader (Sajini & Kallimani, 2021). Compared with traditional news, content on social media might contain more unverified content. Since there are easy to post and share information on social network, people may share the content without fact-checking or verifying the source. Social media information seems to be lack of credibility.

People trust information on social media without vigilance may have a severe impact. First, disinformation may be published on social media to intentionally, like taking advantage in the political environment (Pierri & Ceri, 2019). If people do not verify the information, people's options appear to be easily affected by evil intentions. Moreover, when misinformation spreads on the internet, the information may lead to misunderstanding and cause serious consequences like causing panic during covid-19 mentioned in the survey (Ahmad & Murad, 2020).

1.2. Problem Statement

Social media create a place for people to share their life, and also create a place for someone to spread fake content intentionally. Some people with bad intentions could spread the fake content to benefit them or harm others.

However, some people lack awareness of fake content and trust information on social media without fact checks since the complicated work. The verification of the information of traditional news organizations seems to be cost much effort. Although some organizations like google provide fact-check tools that help users check the topic's credibility, it is needed to have a convenient way to help readers classify the fake content while not losing the comfort of using the social media platform.

1.3. Objectives

In order to raise raising awareness of fake content on social networks, this project aims to create an efficient way to help people verify fake information on social networks. Therefore, this project would have several objectives want to achieve:

• Verify fake content

• Analysis feature of fake content

• Rising awareness of fake content

First, this project will create a tool that could help social media users classify the content they receive on the platform. It will classify the text content by machine learning and deep learning technology. Apart from the classification method, the tools should be well-designed for the public to use to assist users getting useful information from the social media.

Second, it will create a dashboard to analyze the fake content. It aims to create a convenient way to analyze false content on the internet and discover fake content in

social media.

Finally, it is aimed the system, which including the verify tool and dashboard, could promotes the importance of fact-checking. It would reduce the chance of spreading misinformation and raise the general public's awareness of false content on the internet.

1.4. Scope

The scope of this project could divide to three-part, content detection, analysis dashboard application, and chrome plugin.

For the content classification, although different media types like images, video and text are available on the social network, this project shall focus on text data from social media and news. Regarding social network services, this project shall focus on Twitter. The data may involve another type of data, such as news or other social media content. Nevertheless, it shall not be the focal point of this project. This project shall create an analysis dashboard and chrome plugin for the user. It shall implement the content classification method and help the user to classify the content.

The analysis shall focus on data in the dataset. It should show the information of the dataset. It would be nice to have the latest data on social media since it depends on social media services. It is uncontrollable and not the focal point of this project.

2. Literature Review

2.1. Fake content on the internet

The fake content on the internet usually appears as fake news—this misinformation which may cause misunderstanding to the reader. Misinformation represents information which may not be accurate or mislead the reader (Pierri & Ceri, 2019). It may not spread by intention. Most of the cases are due to human error. Compared with it, disinformation is information spread intentionally to deceive the reader or gain an economic advantage (EUROPEAN COURT OF AUDITORS, 2020).

2.2. Prior method of fake content detection

The fake content could be in a different form of data, including text, media like images, video or audio (Carter et al., 2021). Different types of data have their own preprocessing and detection methods. The text data, which this project focuses on, will be preprocessed by NLP techniques like segmenting the sentence into tokens and allowing the machine could work on calculating the result (Kadhim, 2018).

The detection would be a classification task with three common approaches: feature-based, knowledge-based or learning-based (Mohtaj & Möller, 2022).

2.2.1. Feature-based classification

Feature-based classification use machine learning like decision tree, support vector machine (Khanam et al., 2021) or neural network such as LSTM (G. Purna Chandar

Rao & V. B. Narasimha, 2021). It extracts the feature or pattern from the text. It might also include the metadata, such as the feature of the user who sent the message to the internet (Buntain & Golbeck, 2017). This method using an algorithm may not result in high accuracy. It relies on other statistical values as a feature to improve its result (Khanam et al., 2021). Despite the accuracy, this method is more straightforward in the modelling so it could be a baseline for the detection problem.

2.2.2. Knowledge-based classification

Second, the knowledge-based classification concerns the relationship between entities. It finds the related word or topic of specific terms as a knowledge entity (Jingbo & Tianshun, 2002) and stores the words in a knowledge graph. It uses an algorithm to show the pattern to classify the text (Koloski et al., 2022). The incompleteness of data is a challenge since it has little confidence in finding the relationship to do classification (Chen et al., 2019). This approach allows us to determine the result by considering the link between different words. However, it is hard to cover all word and require a consistent update to the knowledge graph. Therefore, this approach usually relies on third-party services.

2.2.3. Learning-based classification

Third, the learning-based model represents some deep learning models. The transformer model is an example of a learning-based model. BERT is a transformer model with an attention mechanism that could provide more features for classifying text to be more accurate in some cases (Wu et al., 2020). This approach considers the context and

importance of each word, which require more computation power but are more accurate in the result.

2.3. Natural Language Processing

Natural Language Processing (NLP) is a branch of computer science. According to IBM Education, it is processing the text and human spoken language in computer (Education, 2021). It could divide to several sub-problem, Tokenization, Part-of-speech assignment (POS tagging), word/phase order and synonymy are the example of the task of natural language processing (Nadkarni et al., 2011). According to the Nadkarni, it usually performs in a pipeline which chaining different NLP task step by step.

2.4. Text Classification

Text classification is a task to classify the text under a defined class. It has several common steps, which are: read document, tokenize text, stemming, delete stop words, vectorizer, feature selection and apply learning algorithm (‘Text Classification: A Recent Overview’, 2005).

2.4.1. Machine Learning Approach

Machine learning approach is a featured-based classification, it could divide to supervised learning, un-supervised learning, and semi-supervised learning. Both of these types can solve the text classification problem. For supervised learning, in research from Thangaraj and Sivakami, it could be further divided into parametric classifier like logistic regression, naïve bayes classifier and non-parametric classifier

like decision tree and neural network (Thangaraj & Sivakami, 2018).

According to new research, support vector machine, logistic regression and randome forest are most frequently to use, since it is solid, having good performance with small amount of parameter (Haralambous et al., 2019). These method could solve the text classification problem efficiently.

2.4.2. Deep Learning Approach

For the deep learning approach, according to Q. Wang et al., both convolution neural network (CNN) and recurrent neural network (RNN) can be applied as a deep learning approach. However, it also states that CNN cannot consider much information and RNN can perform better with the context (Q. Wang et al., 2021).

A part from the traditional neural network, transformer model is a new solution which applied the neural network and pretrained with massive amount of data. It is a new architecture for natural language. Accord to research of google, the encoding layer of the transformer model will consider the context of the text fragment, which called self-attention mechanism. (Transformer: A Novel Neural Network Architecture for Language Understanding, 2017). It is also an approach to solving the text classification problem.

3. Methodology

3.1. Overview

This section introduces the requirements and design of the system. This section focuses on the use cases and technical components as opposed to the implementation specifications. The system has been designed based on the requirements and the software engineering principles by established best practices. The system's primary objective is to provide fake content classification and dashboard analysis to its users. The details of these features will be explained in the following section.

3.2. Requirement

3.2.1. Functional requirement

3.2.1.1. Fake content detection

Users should be able to do fake content detection on a specific text. The web application should provide a detection function to the user and return a score on the provided text, which indicate whether the text is real or fake.

3.2.1.2. Web dashboard analysis

The user should able to analyze the data from the dashboard, and the system should aggregate and transform the data and display it on the dashboard in graph, diagram or table form.

3.2.1.3. Chrome plugin for fake detection

Users should be able to use a chrome plugin to perform fake content detection on selected text. The user selects the text and triggers the detection on the context menu.

3.2.1.4. Non-functional requirement

Human-centered design

The design should be human centered, which should be designed to start from the viewpoint of the end-user.

Maintainable

The system should be designed in maintainable ways, such as applying applicable design principles. So that to minimize the maintenance cost.

3.2.1.5. Technical requirements and specification

Three tier architecture

The system should apply the three-tier architecture, which separates the system from data, business logic and user interface.

Ant Design System

The user interface should apply the Ant Design system, using the design language and values in this system to make the user interface consistent and user-friendly.

3.3. System usecase

The requirement has specific content of the system needs. It can breakdown it into a use case.

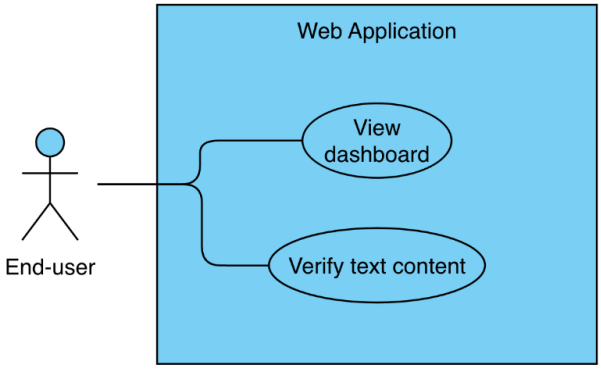


Figure 1 Use case diagram

3.3.1.1.Use Case Specification

Use Case ID:	UC 1	
Use Case Name:	View Dashboard	
Actor(s):	User	
Description:	View the dashboard at web page	
Trigger:	This use case is initiated when the user requests the web page.	
Normal Flows:	**Actor Action**	**System Response**
	Step 1: Request the web page	Step 2: Server aggregate the data from database

		Step 3: Return data to client-side web page Step 4: Client-side render chart and other view component to display the chart
Post-conditions:	Show the web page with analysis dashboard.	

Use Case ID:	UC 2	
Use Case Name:	Verify text content	
Actor(s):	User	
Description:	Verify whether a text content is true	
Trigger:	This use case is initiated when the user click send a request of verifying a text. The request may make by clicking a button.	
Normal Flows:	**Actor Action**	**System Response**
	Step 1: User make request with text content	Step 2: Server call the detection model to classify the text Step 3: Return classification result Step 4: Client-side render corresponding result to user
Post-conditions:	Show the result in web page	

3.4. System structure

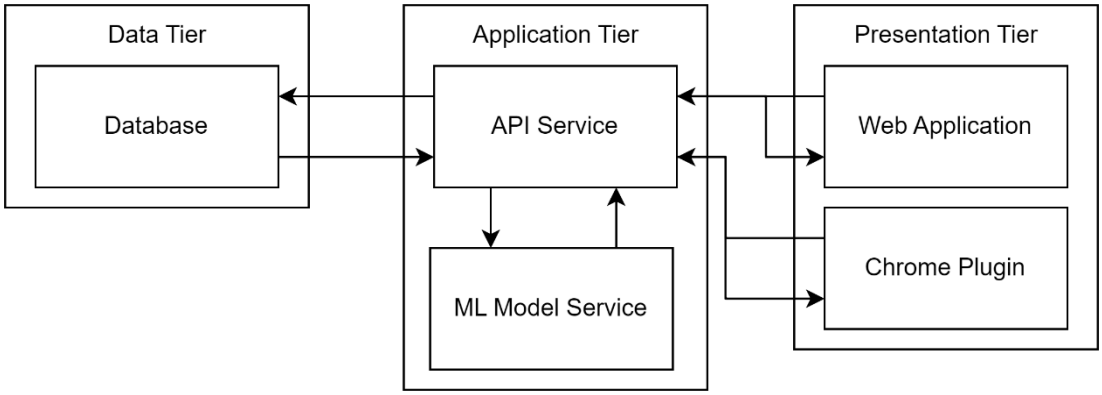


Figure 2 System architecture diagram

The system is designed to construct into three tiers, which could be divided into five components. It would apply three-tier architecture, which divides the presentation tier, application tier, and data tier. The three-tier design could separate the development. Applying changes to one system will not affect another, except breaking changes (*Three-Tier Architecture*, 2021).

The presentation tier includes two components: a web application and a chrome plugin. It provides a fake content detection service and dashboard for end-user to analyse. The web application should contain text input for fake content detection and a dashboard.

The Application tier also includes two components: a server responsible for the application programming interface (API). It should aggregate the data and return it by API. Another server will work on a machine-learning service that detects fake content. It will provide an API for performing machine learning model influence on request.

Finally, the data tier contains a database which stores the data for analysis. The data would be dataset data and other data scraped from a website like Twitter's tweets.

3.5. Technical component

This section will introduce the technology expected to use in the above design. There are some frameworks or applications which could help to construct the system. The exact reason will be explained below. Here is a summary of the technology used in each component.

Web Application	**React.js**
API Server	Express.js
Model Server	Flask
Database	MongoDB

Table 1 Technical component

3.5.1. Web application

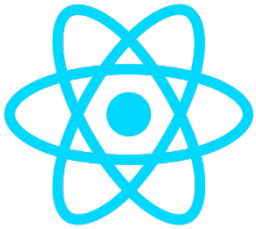


Figure 3 React.js logo (*React – a JavaScript Library for Building User Interfaces*, n.d.).

The web application will be developed in React.js. It will be a separate web application,

and the data will be retrieved from the API service. It will be implemented in React.js. It is an open-source JavaScript framework developed by Facebook. It is a component-based framework which divides each component. It could allow each component's attribute to be encapsulated and make constructing a complex user interface easy (*React – a JavaScript Library for Building User Interfaces*, n.d.).

Compared with other technology, like Vue.js and Angular, React.js are component-based, allowing to create of functional components. It could allow easy reuse of the component in multi places.

TypeScript will be used to develop React.js. It enables type-safe so that it can avoid some run-time errors. It could help to handle large data structures, which prevents messing up with different attributes.

3.5.2. API server

Figure 4 Node.js icon (Node.js, n.d.)

Figure 5 Express.js logo (Express - Node.js Web Application Framework, n.d.)

The API server will be developed in Express.js. It provides API services that aggregate the data from the database. Also, it connected the service between the model service and the front end. Express.js is a framework to establish web applications quickly. It is

suitable to do the API server since it is lightweight, which helps the response quickly and has flexibility to make the system which allows a fully customized structure and easily applies the software engineering practice into it to ensure the codebase quality. It will also use TypeScript to develop, and the reason is the same as above.

3.5.3. Model server

Figure 6 FastAPI logo (FastAPI. (n.d.). https://fastapi.tiangolo.com/)ss

The model server will be implemented using Fast API, a Python web framework suitable for developing small web applications. The model server will only run the machine learning model. The model is compatible with running in Python and does not require other features. Therefore, a lightweight server is needed for the model, and Fast API is suitable for this job.

Since the model requires a lot of performance, it is separated from the API server to help allocate resources more easily.

The model server using Flask should also apply type annotations (Typing — Support for Type Hints, n.d.). This application could help Python have a type-safe feature.

3.5.4. Database

Figure 7 MongoDB icon (MongoDB, n.d.)

The database will use MongoDB, which is a NoSQL database. The database should store the dataset data, and the data scraped from the website. These data are mainly the social network feeds and may need to be better structured.

MongoDB is a document-based NoSQL database, and it stores the data in JSON style, so it is suitable to store the flexible structure social media data.

3.6. Model training

Apart from the software architecture, the machine learning model would be developed. Several models would be trained and compared to their performance at the end. A typical machine learning model will be trained as a baseline model for further evaluation. Some advanced models, like the Word2Vec model, which calculates word similarity, would also be applied. Finally, the deep learning transformer model would fine-tune the downstream task for fake content detection. The package sci-kit-learn, Spacy, Gensim and Hugging Face is considered to conduct the data preprocessing and model training.

Figure 8 Scikit learn logo (Scikit-learn: Machine Learning in Python — Scikit-learn 1.2.1 Documentation, n.d.)

Scikit-learn is a Python machine-learning library that implements many machine-learning algorithms. It helps to build the model in a short period.

Figure 9 Spacy logo (spaCy · Industrial-strength Natural Language Processing in Python, n.d.)

Figure 10 Gensim logo (Gensim: Topic Modelling for Humans, n.d.)

Spacy is a natural language processing (NLP) library which provides tokenization, tagging, and word vector functions. Gensim is a topic modelling library. Using these two package libraries could help to build a Word2Vec model. These two libraries are designed to handle a large amount of text and provide reasonable data processing speed.

Figure 11Figure 11 HuggingFace Transformer Logo Hugging face logo. (n.d.).

https://huggingface.co/brands

Hugging Face is a community which provides a package to perform transformer downstream task training. The community provide the pre-trained file of a transformer model. It allowed us to build a transformer model on a specific task quickly.

The model training task would be a text classification task since it will do the fake content detection. The content would be text, and it will predict whether the text belongs to accurate information or false information, which is a classification. The text needs to do preprocess before building the model. It needs to be transformed into text, including tokenization and word embedding.

3.7. Deployment

To complete the whole implementation process, deployment is considered to do in this project. It is aimed to deploy the application on cloud, such as Amazon Web Services (AWS). Some AWS services is suitable to use in this project.

AWS ECR and ECS Cluster allow the project to deploy a containerized server into cloud. AWS SageMaker allow to deploy machine learning and deep learning model and running as a serverless services. AWS S3 bucket could be used to store the model file and static file.

Apart from the AWS, the front-end Reacts is a client side render web application, which only need a place to host the file, GitHub pages is a suitable platform to host the web page in a cost-effective ways. Also, the database will be separate from the AWS cloud, it can make use of the official services provided by MongoDB called atlas, which is a cloud mongo DB ssstorage allow running as a serverless services.

3.8. User activity

The system has two components faced by end-users: the web application and the chrome plugin. Both of them would perform text detection. Here is the activity workflow of the application.

3.8.1. Web Application

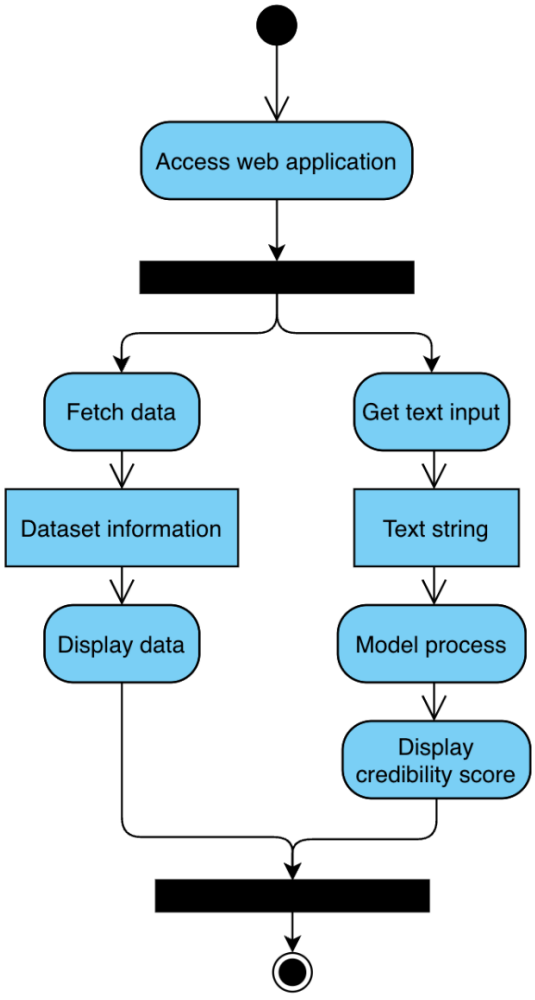


Figure 12 Activity diagram- web application

The web application would display a web page for the user to browse. It will fetch data from the server and display dataset information on the web page. Meanwhile, the user could input the text string to do fake detection. The web page will display the credibility score once the model influences the text.

3.8.2. Chrome Plugin

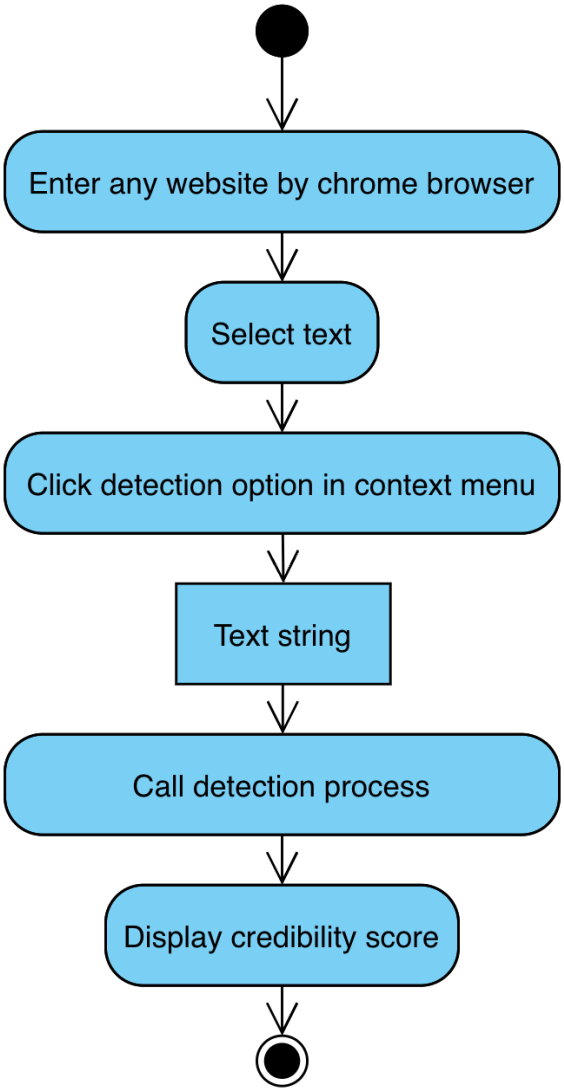


Figure 13 Activity diagram - chrome plugin

The chrome plugin is similar to the activity of a web application. The plugin will add an option to the context menu. It allows the user to select text. After that, the user could select the option in the context menu. The plugin will call the model process through API and display the credibility score to the user.

4. Implementation

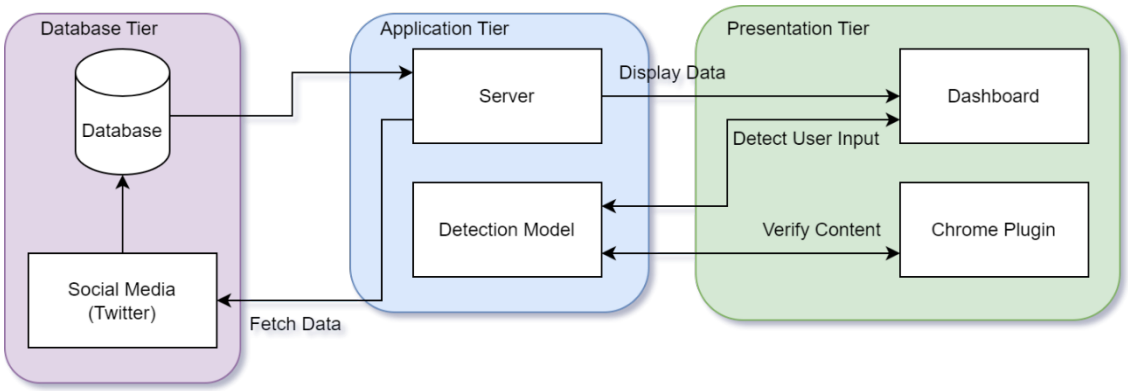


Figure 14 System overview

4.1. Overview

This section delves into the design and implementation of the system. The system comprises multiple components that will be thoroughly analyzed concerning tasks and operational flows. In addition, it will examine the information processing performed on the dataset and the considerations made during the implementation stage.

4.2. Backend

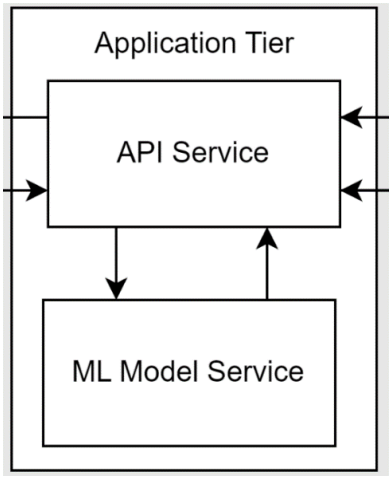


Figure 15 Backend architecture diagram

The system's backend consists of two components: the API server and the machine learning model service.

The API server has been designed following the RESTful API standard and is responsible for handling all requests from the web application. It supports the following APIs:

HTTP Method	End point	Params	Remark
GET	/v1/model/available		Show available model
POST	/v1/model/detect	text: string, model: string	Detect the string

Table 2 API Server endpoint

There is also other endpoint for front-end to fetch the data for dashboard data display.

The model detection service only works on model detection. Here is the API supported:

HTTP Method	End point	Prams
POST	/v1/detect/{embedding}/{method} e.g. /v1/detect/tfidf/svm	text: string

Table 3 Model server endpoint

In the implementation of the API server, the Model-View-Controller (MVC) pattern and the Repository pattern have been adopted. Despite not having a view for the user, the API server utilizes API responses to present the information. The responses, business logic, and data model have been defined separately to maintain a clear separation of concerns. The Repository pattern has also been applied to separate the data connector from the controller and data model, making the code more modular and maintainable. This modularity also facilitates easier testing.

4.3. Frontend

The front end would composite the chrome plugin and the web application dashboard.

React.js would implement the web application. It would be component based. Each component would control its state. The component would be designed to be reusable, like the pie chart detection input box.

There will be different pages in the web application. Here is a wireframe design of the system (the final design will be different, here, only indicate the simple design of the component).

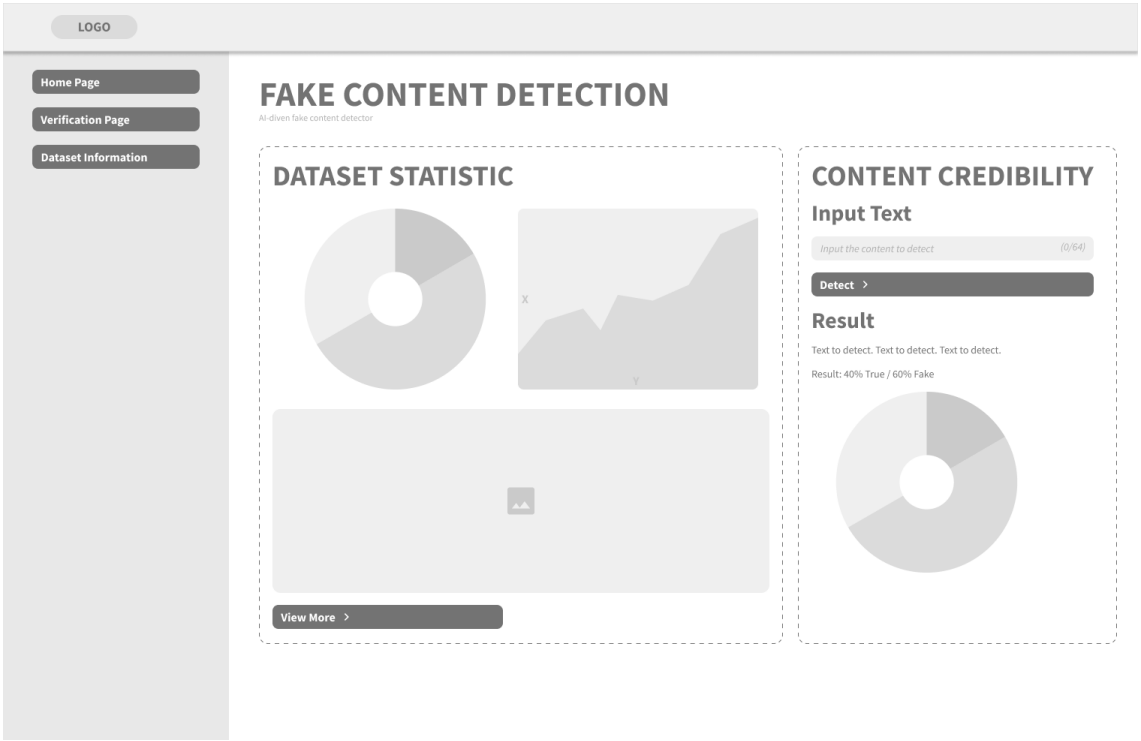


Figure 16 Wireframe - Home Page

First, there will be a home page containing two parts: dataset information and content

detection. The dataset will aggregate some simple information like label distribution and word frequency. Also, a word cloud would be generated at the page's bottom. The detector allows the user to parse the text to detect fake content. The user could view the result in a pie chart which indicates the credibility score.

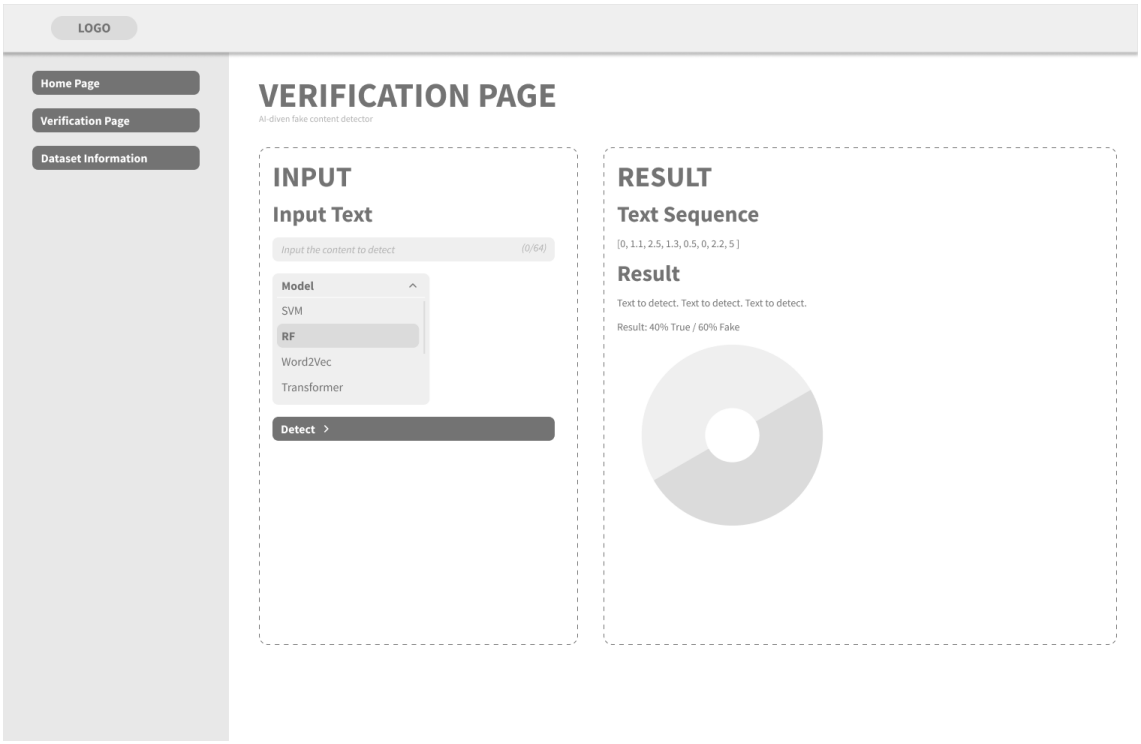


Figure 17 Wireframe - Verification Page

Second, there will include a verification page which performs text verification. It provides more functions than the home page, allowing users to select a specific model for detection. The output also would be shown for the user to evaluate the result.

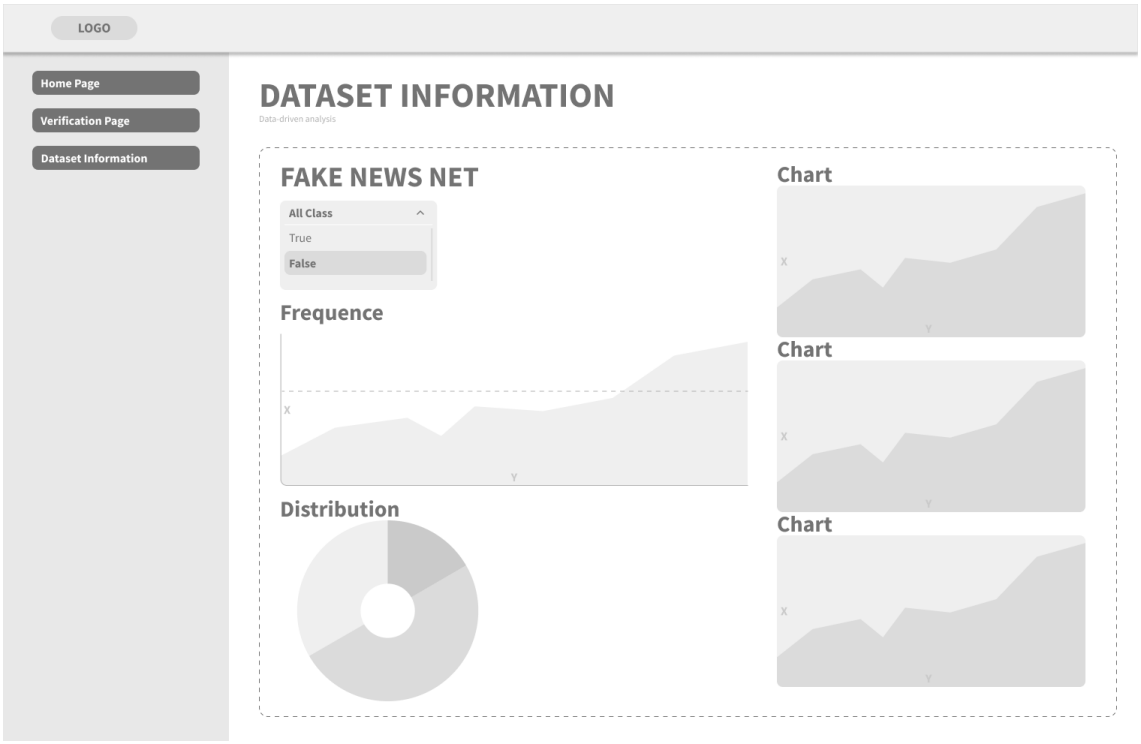


Figure 18 Wireframe - Dataset Page

Last but not least, dataset information will be provided. The dataset information page would provide more information than the home page. Allow users to do simple filtering to perform simple analyses of the dataset.

For the chrome plugin, the option would insert into the context menu. It will call the API service to get the detection output from a specific text.

4.4. Data acquisition

4.4.1. Data Source

This system uses two datasets: the FakeNewsNet dataset and the LIAR dataset. The FakeNewsNet dataset is chosen as the primary dataset for model training due to its large

size and relevance to the topic, which is sourced from social media. On the other hand, the LIAR dataset, although relatively small, will used as a testing dataset to evaluate the system on unseen data. It is worth noting that the nature of the data in the LIAR dataset may differ from that of the FakeNewsNet dataset, making it an attractive option for evaluating the system's performance on new data.

4.4.1.1. FakeNewsNet

This dataset contains news and social media content. The content came from a fact check reporting websites Gossip Cop and PolitiFact. Both news data and social media data come from these two websites. The raw dataset format is the following

```
{
    id: ""
    news_url: ""
    title: ""
    tweet_ids: []
}
```

The data contain tweet_ids instead of the tweet content. So it is needed to fetch the data by id from the Twitter API. The fetching script is developed and stores the raw response file as JSON.

The package tweeps are integrated with the Twitter API, which helps fetch the data quickly. Using the Twitter tweets look-up, it supports fetching 100 tweets at one API call. Each tweet is stored on a disk. Each data contain the tweet, the author of the tweet, the metrics, also the named entity of the tweet by twitter.

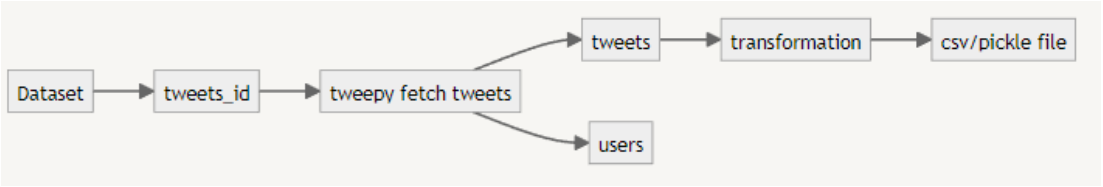


Figure 19 FakeNewsNet data process flow chart

After fetching data from Twitter, the data is transformed into a different shape, aggregating all the data into pandas data frame format and then saved as CSV and pickle file. Since the pickle file is binary, it could decrease the file size and increase the loading speed of the data.

The final dataset is transformed into only two rows: text and label. The label is transformed into 0 and 1, which represent false and true, respectively,

	text	label
0	What's Really Going on Between Scott Disick an...	1
1	@IanBouillion ya but you're way more interesti...	0
2	*kourtney kardashian voice* literally mom you'...	0
3	Tonight we are awarding Sustainable Nation Ire...	0
4	7/23/18: White House Press Briefing - The Tim...	1
...	...	...
1368182	Total Bellas Recap: The Family Takes a Trip to...	1
1368183	Fall out boys and wiz concert and trey songs ...	0
1368184	@FoxBusiness @Jim_Jordan NOW you're going to s...	1
1368185	@dutkae Flip or Flop Katy?	0
1368186	Millie Bobby Brown and Gaten Matarazzo Rally f...	1

Figure 20 FakeNewsNet sample data

From the dataset, it is possible to see some statistics of the dataset. The dataset contains 1368187 rows, of which 932369 are true, and 435818 rows are false data.

Label	Count

Fake twitter post	684473
Real twitter post	1276204
Fake news content	5755
Real news content	17441

Table 4 FakeNewsNet dataset size

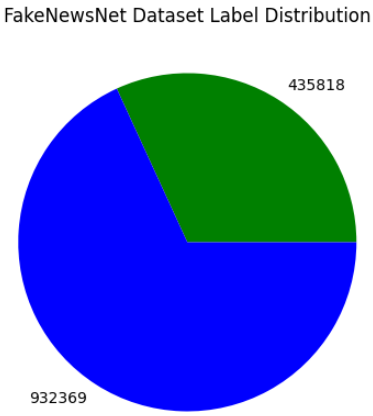


Figure 21 FakeNewsNet label distribution

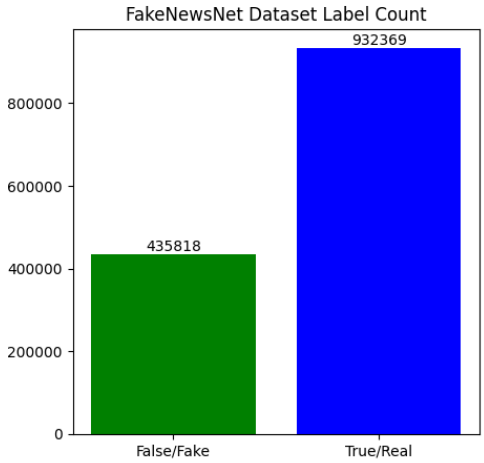


Figure 22 FakeNewsNet dataset label count

From the above diagram, it is possible to observe that the label is imbalance. The true labelled data is two times the false labelled content. This may negatively affect the machine learning model training, which produces a bias in the prediction. It will bias toward the true label instead of the false label. However, since there is a certain amount of labelled data, it is not a massive concern to the model performance. It could fix by controlling the training data.

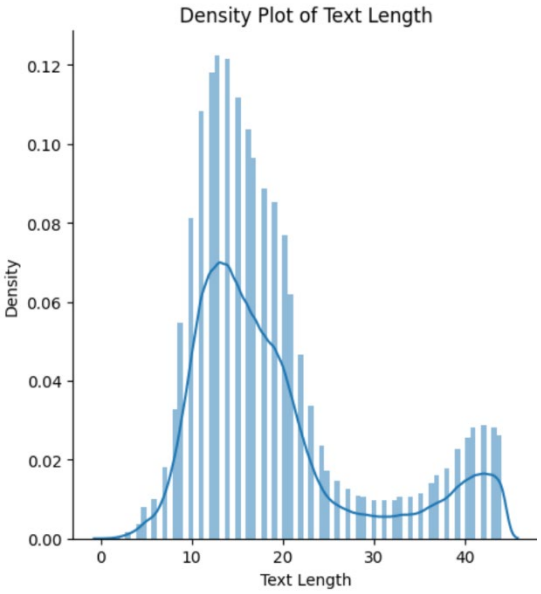


Figure 23 FakeNewsNet text length

Furthermore, to analyze the text content of the dataset, the text length of each text content is processed. Most of the data are around 10 to 20 words. Thus it can be classified as a short-text classification. This could lead to a significant problem, as the context of the text may need to provide more information for classification.

Furthermore, to analyze the text content of the dataset, the text length of each text content is processed. Most of the data are around 10 to 20 words. Thus it can be classified as a short-text classification. This could lead to a significant problem, as the context of the text may need to provide more information for classification.

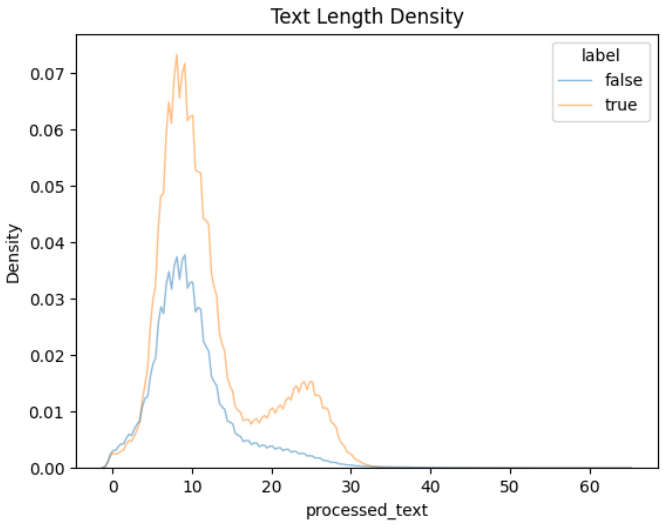


Figure 24 Text length distribution of two label

By separate the label to plot the density, it is able to observed that the text more than 20 word most likely is true. Also, most of the false data are under 20 words.

Metadata fields were returned from the Twitter API, including context annotation, entity, domain, entity type, created time, user information, and metrics. An exploratory data analysis can be conducted on the dataset.

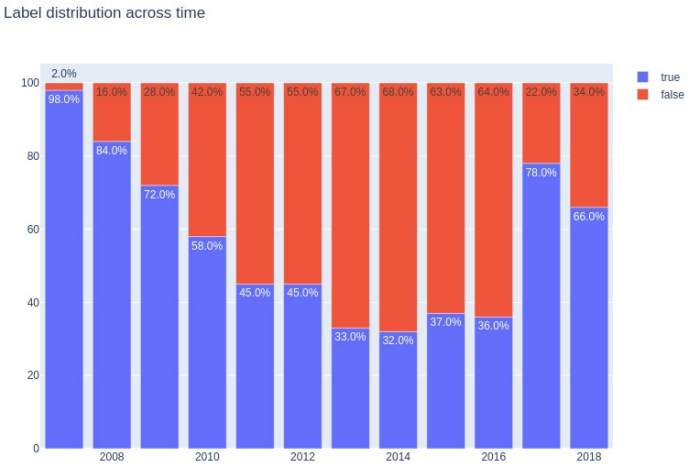


Figure 25 Time series label distribution

First of all, form the time series distribution, it is able to observe the data in 2013 to 2016 is relative more false content in the internet, which having more than 60% of data are false content. However, this may not reflect that the society having more false content in that period, since this may relate to the data collection method of the dataset.

Figure 26 Word Cloud of the dataset

By visualizing the word in the word cloud. It can observed that “new” and “news” are the most largest word that having most occurrence.

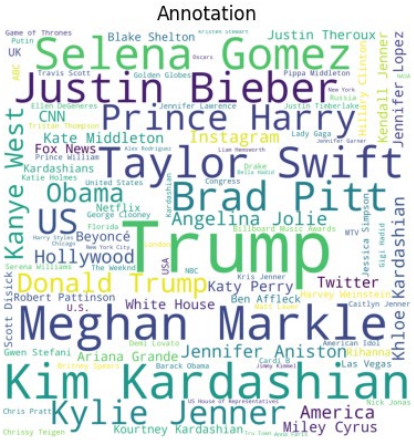


Figure 27 Dataset text annotation word cloud

The data having the annotation field, which represent the content of the text. using the text to form a word cloud is able to analysis the word frequency. It is able to see that the word “Trump” occur many times, which is the biggest word. The word ‘Kardashian” follow as the second largest. With a look of the large size word in the graph, it is able to see that most of the data contain person in the data.

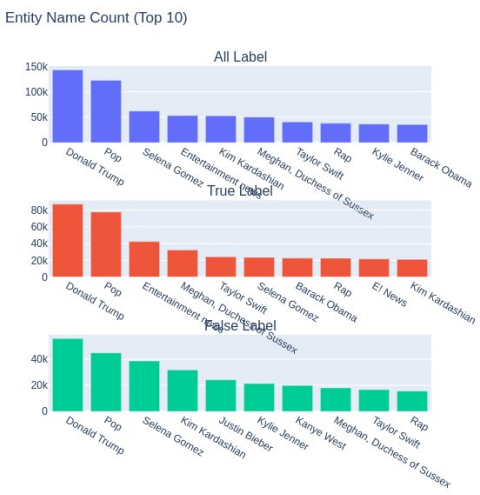


Figure 28 Top 10 entity name

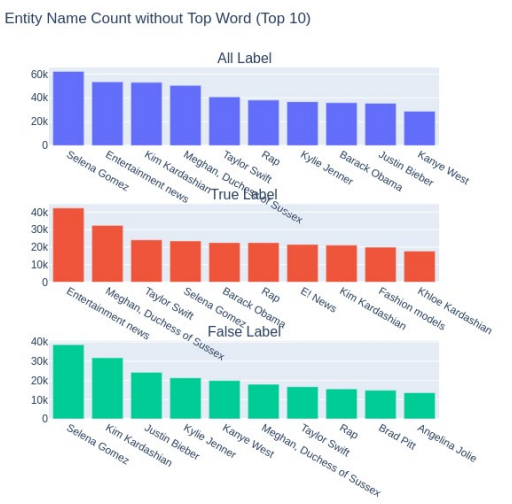


Figure 29 Top 10 entity name without top word

The annotation could divide to two type which is entity and domain name. It is able to see that there are some word across true and false label are the same. Therefore, by filter the same word across different label.

Form the entity count, the entertainment are most likely to be a true, this may because it purpose is for entertainment which may less involve the misleading information. For the false, label, the entity “Selena Gomez” and “Kim Karadashian” who are the famous people around the world, the entity following these two items are also the person. This show a record related person may have chance to be a false content.

Figure 30 Top 10 domain name

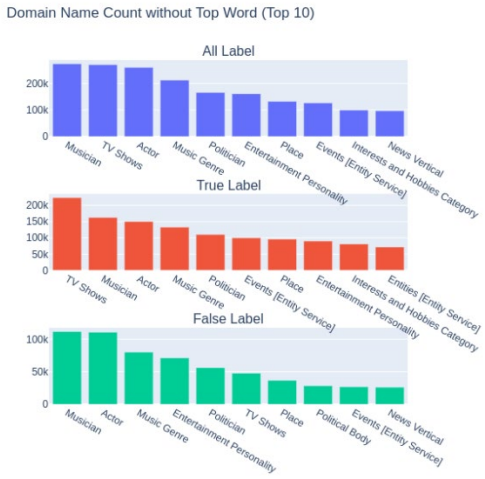


Figure 31 Top 10 entity name without top word

For the domain, it show the content domain. It can clear see that most of the true label are related to TV shows and false label are related to musician and actor. This could help for a simple classification by the content of the detected text.

Figure 32 True tag word cloud

Figure 33 False tag word cloud

In twitter, tag is an important feature to attract people and promote the tweet. The tweet usually includes the related topic of the text content. By separate the true and false label to analysis the word cloud. It can see there are quite different between two label.

From the tag of true label. The entertainment standing out which like the above mentioned. Another worth noting is the content contain news may also likely to be true data, some word contain news like “FoxNews”, “Breaking News” are related to news and count as news. For the false label, some word like “WakeUpAmerica” is false label, which show some world related to political issue may count as fake content.

4.4.1.2.LIAR dataset

LIAR dataset (Wang, 2017) is a public dataset that collected a decade of data. The dataset is formed by news content. This dataset is already split with training, validation and testing datasets.

Label		Count	
Real	True	2466	4529
	Mostly True	2063	
Fake	Half True	2638	8307
	Barely True	2108	
	Pants on Fire	1050	
	False	2511	

Figure 34 LIAR dataset information

It has six labels, which are “True”, “Mostly True”, “Half True”, “Barely True”, “Pants on Fire”, and “False”. Only “True” and “Mostly True” are counted as “True” content with 4529 records. Other labels are counted as “Fake” content with 8307 records. The dataset was labelled by using PolitiFact services.

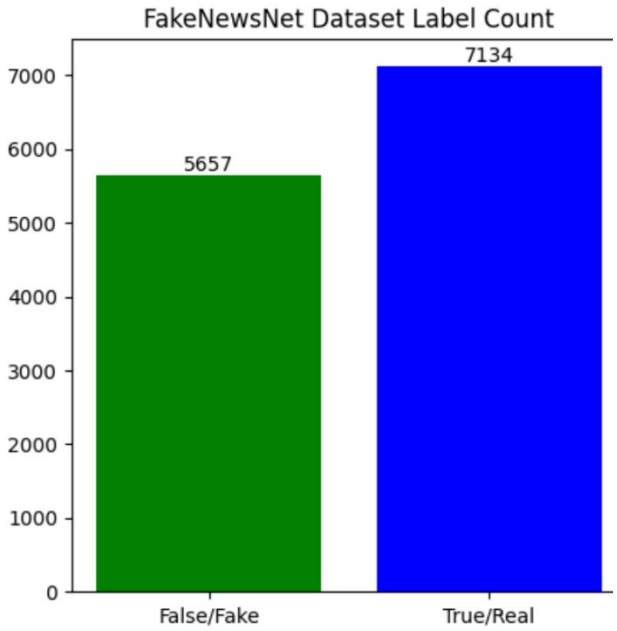


Figure 35 LIAR dataset label count

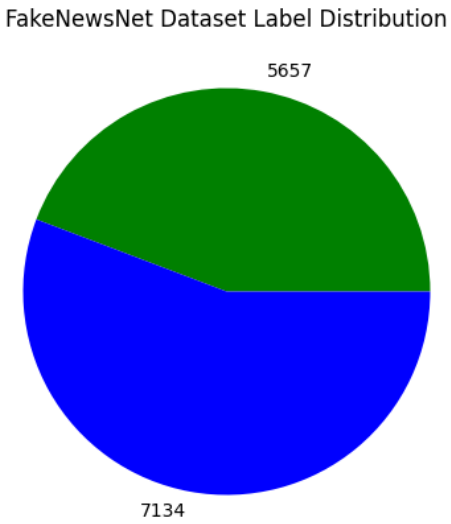


Figure 36 LIAR dataset label distribution

After aggregate the column, the data could be count as evenly distributed.

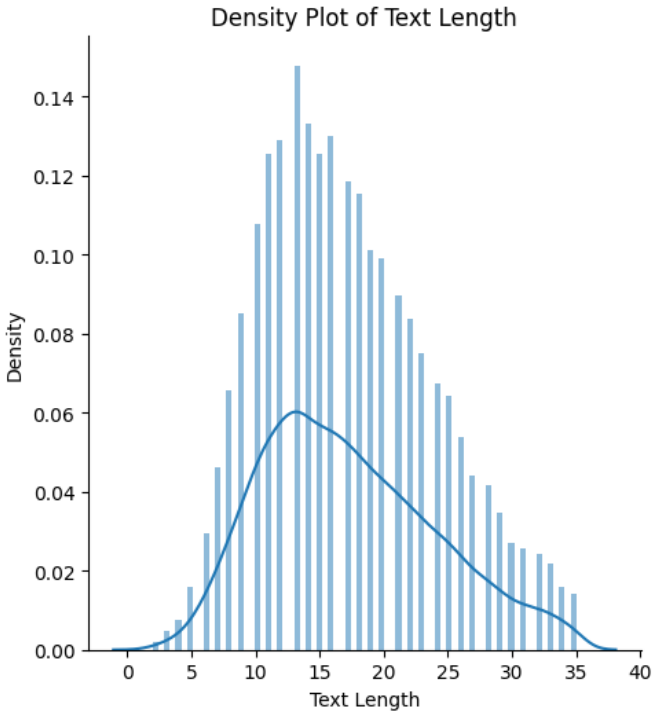


Figure 37 LIAR text length

For the distribution of text length of each data, most of the data are around 5 to 30 words, which is a wide range of data. However, since most of the data are kept on a short length, it is aligned with the shape of the FakeNewsNet dataset. Both of their lengths is short.

4.4.2. Data preprocessing

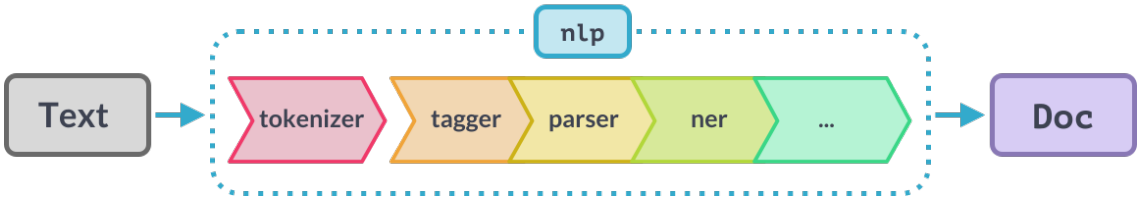


Figure 38 spaCy preprocessing pipeline (Language Processing Pipelines, n.d.)

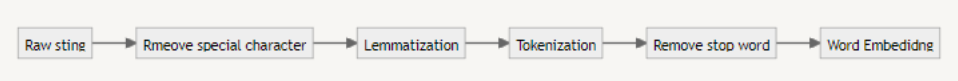


Figure 39 Data preprocessing flow

For general training, the data will first remove special characters, then perform lemmatization, transforming words from different forms into a single form. After that, tokenization is performed, splitting text by space or phrase. Stop words are removed, and the text is ready for word embedding. This process is done using the Spacy library.

For machine learning and deep learning, different models may require slight variations in the processing. The general steps are tokenization and word embedding. For example, the transformer model would do the tokenization using its tokenizer with its pre-trained corpus.

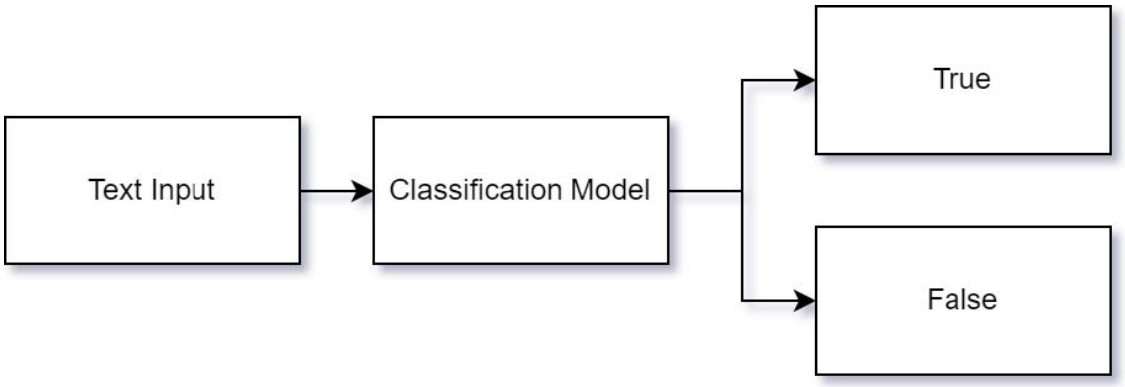
For typical machine learning models, two methods are used for word embedding: TF-IDF and Word2Vec. TF-IDF is calculated using sci-kit-learn and embeds words using their IDF values. Word2Vec uses Gensim, which trains the corpus using the FakeNewsNet dataset. After that, the model is built from the data.

4.5. Model training

The model would be built to detect fake content. The problem would be solved by machine learning. The task and the method would describe in the following.

4.5.1. Problem framing

Framing the fake content detection task as a machine learning problem is a classification task that classifies a text, with a Boolean value as the output. This is a binary classification problem, with two possible classification classes: true and false.



This project will solve the fake content classification problem through machine learning and deep learning. Machine learning is a subfield of artificial intelligence that uses algorithms to detect patterns in data, while deep learning is a subfield of machine learning that applies deep neural networks. It is assumed that the deep learning approach will perform better than the machine learning approach, so machine learning will be used to set the baseline of the performance. Different machine learning and deep learning methods will be compared to find the best-performing model to solve this problem.

4.5.2. Model training

Model training will be built by comparing different methods. Variables such as the word embedding method and different model algorithms will be changed for experimentation.

Bag of words, TF-IDF, Word2Vec and FastText will be used to compare the word

embedding. Bags or words and TF-IDF is a count-based method, it belongs to term frequency embedding method. Bags or words will simply create a dictionary for storing the frequency of the word. TF-IDF will highlighting important words while ignoring frequent ones such as stop words.

Word2Vec and FastText is a similarity model which considers the context of the text sentence, embedding the words by using the similarity between the words and other words in the sentence.

Different machine-learning algorithms will also be compared. Support Vector Machines (SVM) will be considered for text classification, while Random Forests (RF) will also be tried. Decision Tree (DT) and Logistic Regression (LR) also consider to be done the classification task. Naïve Bayes (NB) will serve as the role of probability model in this task. In addition to the machine learning algorithms, neural networks will be considered. Recurrent Neural Networks (RNNs) such as Gate Recurrent Units (GRUs) will be considered for comparison, as RNNs are suitable for sequence prediction, which can help to consider the context of the text.

Here is a test plan on comparing the model.

Word Embedding Method	**Algorithm/ Method**
BOW/ TF-IDF	SVM
	RN
	LR

	DT
	NB
Word2Vev/ FastText	SVM
	RN
	LR
	DT
	NB
	RNN
Transformer	BERT
	XLNet

The model will be compared using metrics such as accuracy. Each model will also be evaluated for overfitting or underfitting and the prediction results. During training, multiple models may be produced with different parameters. The best model will be chosen by evaluating the loss and accuracy.

4.5.3. Evaluation Method

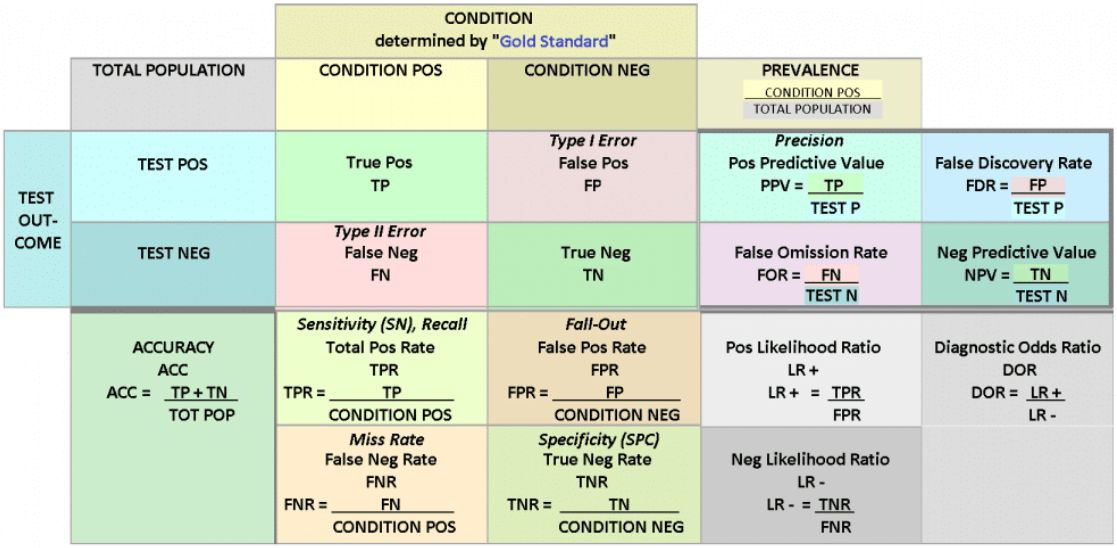


Figure 40 Confusion Matrix (Daniel, 2020)

A key metric is needed to compare the model. Accuracy is a suitable metric which evaluates the performance of the result. It is simple and easy to understand. In advance, the confusion matrix could be applied as the metrics, which the precision and recall rate are helpful to determine the performance of a model. By the nature of the problem, it is better to identify actual content wrongly as fake content than fake content be true content, i.e. It prefers false negative (type 2 error) over false positive (type 1 error). Therefore, this project will optimize the precision instead of the recall rate. In addition, the F1 score also is considered since it combines the precision and recall rate as comprehensive metrics.

4.6. Database

The database is using MongoDB. The data from the FakeNewsNet dataset would be

transformed and imported into the database. The database will then store all the data and provide access to aggregate the data to show in the dashboard.

For the data shape, the details would be shown in the following entity-relationship diagram (ERD).

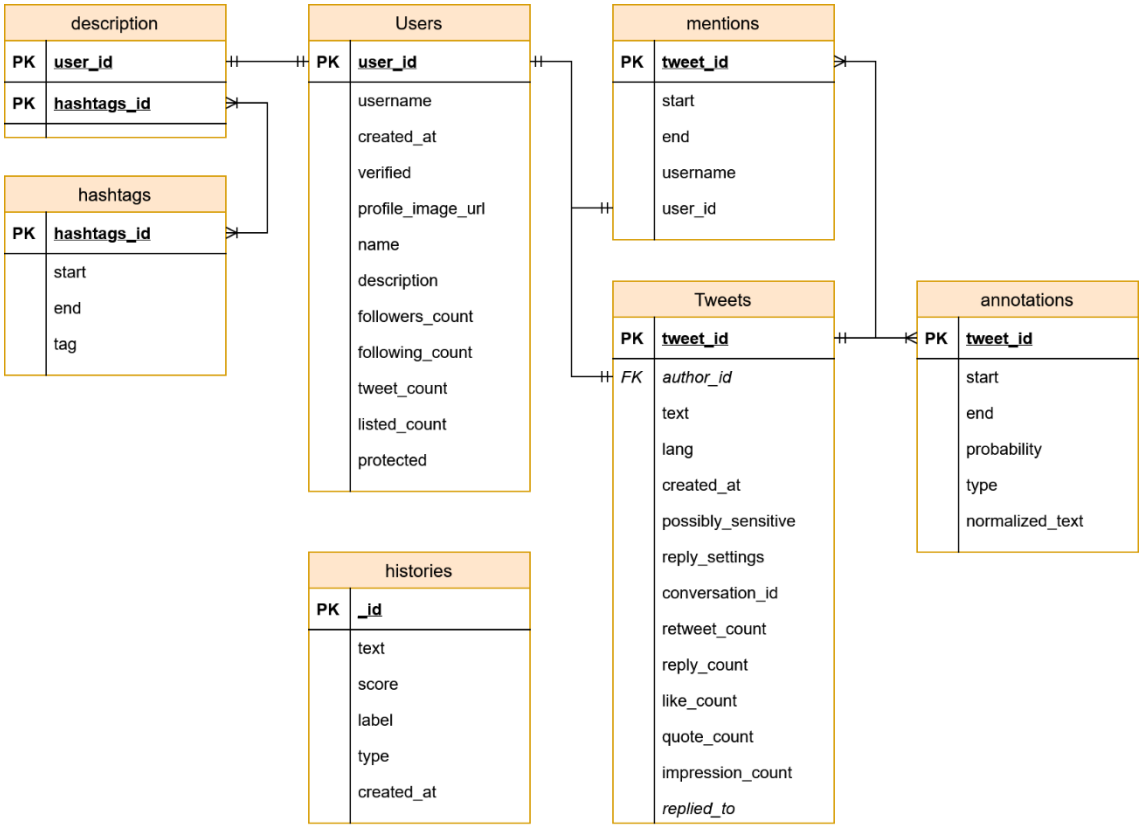


Figure 41 ER diagram

The data is arranged in several entities. They are users, tweets, mentions, annotations, descriptions, and hashtags. The tweets and users are the main entities which are connected by author_id. A user will be the author of a tweet.

The tweets have two entities labeled by Twitter, which are mentions and annotations.

Mentions indicate the user mentioned by the tweet, such as @XXXUser, and annotations indicate the content of the tweets, such as “sports”, “MLB”, “Racing”, and other topics.

The user entities have a field related to descriptions, which shows the hashtag in the user profile description.

Both users and tweets contain some public metrics such as like_count, tweet_count, and followers_count. These fields are some statistics of the data, so further analysis can be done on the data.

Apart from the dataset’s data, a histories entities also included to represent the information of the system detected text.

For the data, there will be a schema to do the schema validation before it is entered into the database. The validation will ensure the data type of the data, so that the data can be consistent.

5. Result and evaluation

5.1. Web Dashboard

The web dashboard serves as a tool for easily using the model and conducting fact checks, while also providing simple analysis on false content. The site map includes three main pages: the home page, detection page, and dataset analysis page. The home page provides an overview of the system's detected text and offers simple statistics about the dataset. The detection page allows users to check the credibility of a given string and view the history of detected records. The dataset analysis page enables users to do interactive filters with labels, see statistics about the dataset, and visualize the dataset metadata and aggregate text content using different diagrams. Overall, the web dashboard offers a user-friendly interface for conducting fact checks and analyzing false content.

5.1.1. Home Dashboard

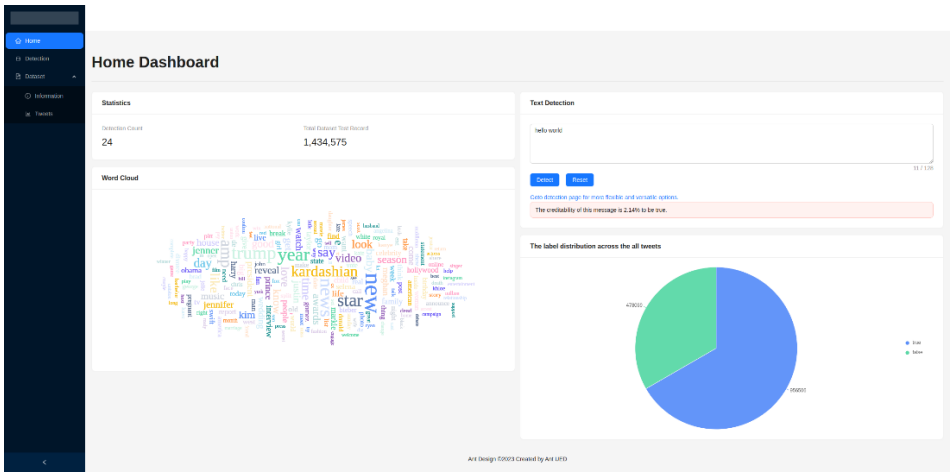


Figure 42 Web dashboard screenshot Home Screen

The home page will be a simple dashboard for user to have an overview. User could view the simple statistic in left hand-side, include the text detected record and dataset record count. User also could perform text detection on the right-hand side.

5.1.2. Detection Page

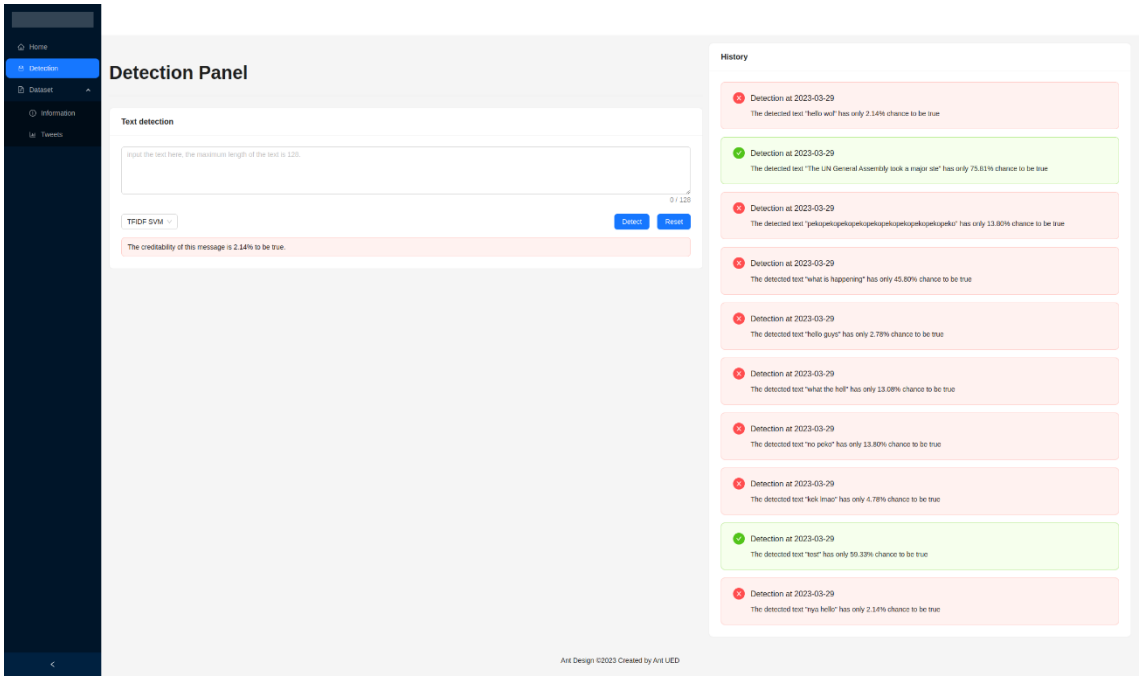


Figure 43 Web dashboard screenshot Detection Screen

The detection page is used for checking the credibility of a given string. Since the model is limited by the token size. Therefore, the detection textbox is also limit by a token size. Apart from the detection function, user is also able to view the history of the detection. The detection history attribute include the detected time, result credibility, used model will be display.

5.1.3. Dataset Analysis

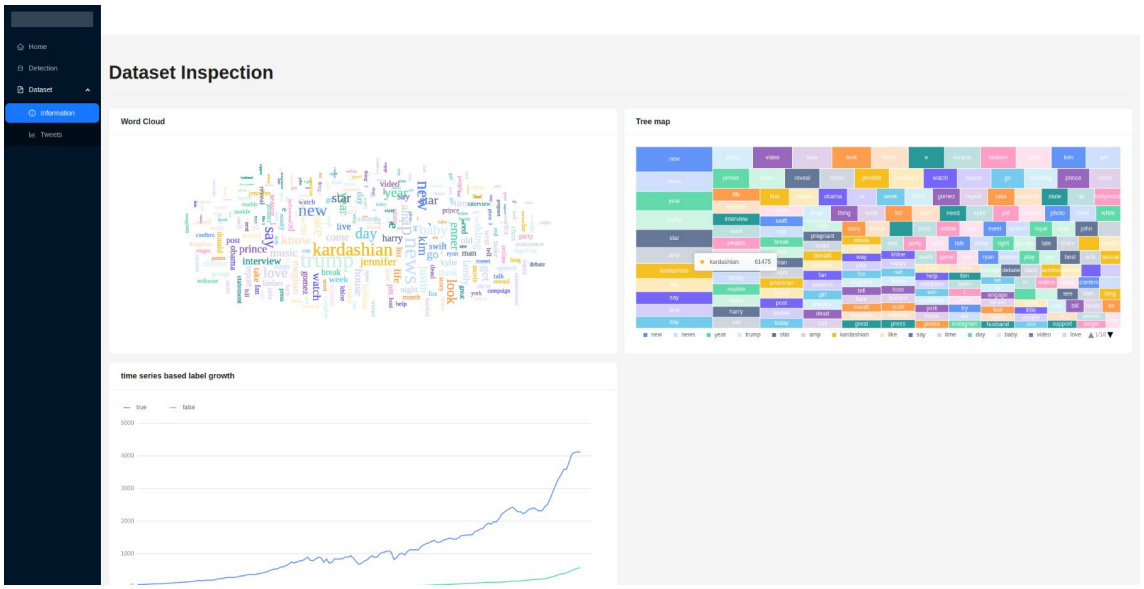


Figure 44 Web dashboard screenshot Dataset Information Screen

The dataset page aggregates the dataset data and visualize it in the dashboard. It show the data for user to do simple analysis with the dataset data.

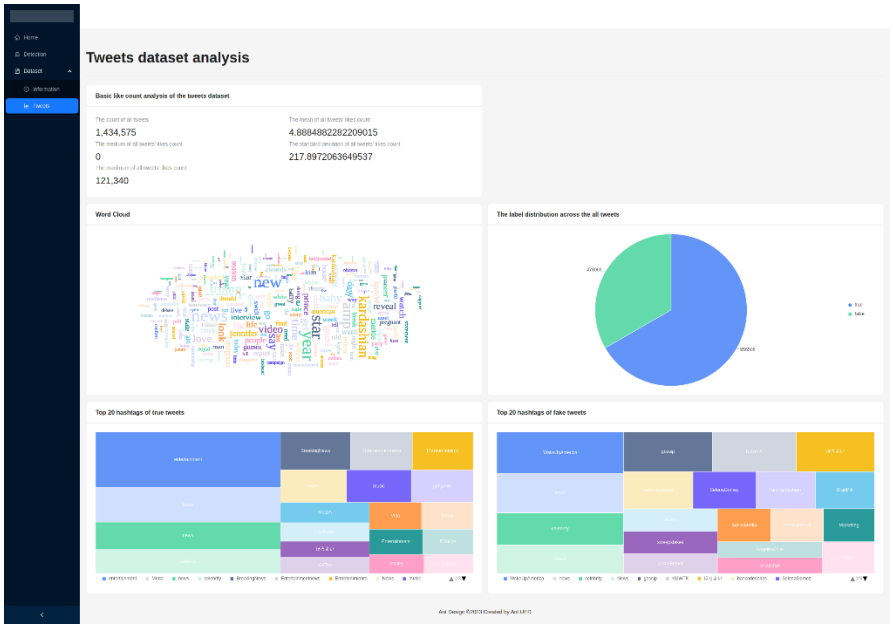


Figure 45 Web dashboard screenshot Tweet Analysis Screen

The dataset analysis page also allow user to do simple analysis with the tweets data, including viewing the statistic of the dataset, and view the distribution of the dataset. A tree map provided for user to view the ranking of the hashtag.

5.2. Chrome Plugin

The web dashboard require user to enter the web site to do the detection. However, it still not so convenient for user to use the model to do the detection. Therefore, a more convenient way, chrome plugin is provided for user to use.

The chrome plugin is inspired from google translate. By select the text, popup button will be shown. Click on the popup button will be send the request to server to perform the detection on the selected text.

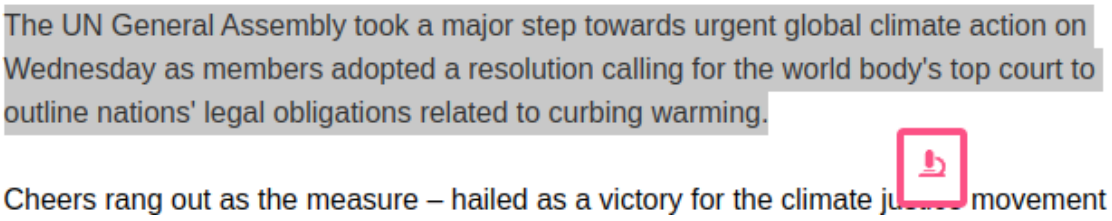


Figure 46 Chrome Plugin popup button

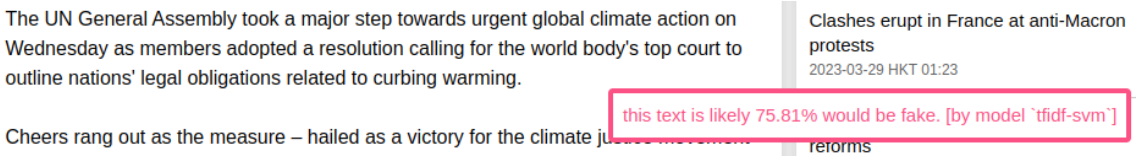


Figure 47 Chrome Plugin Detected Result

The chrome plugin also provide function for user to select the model to use.

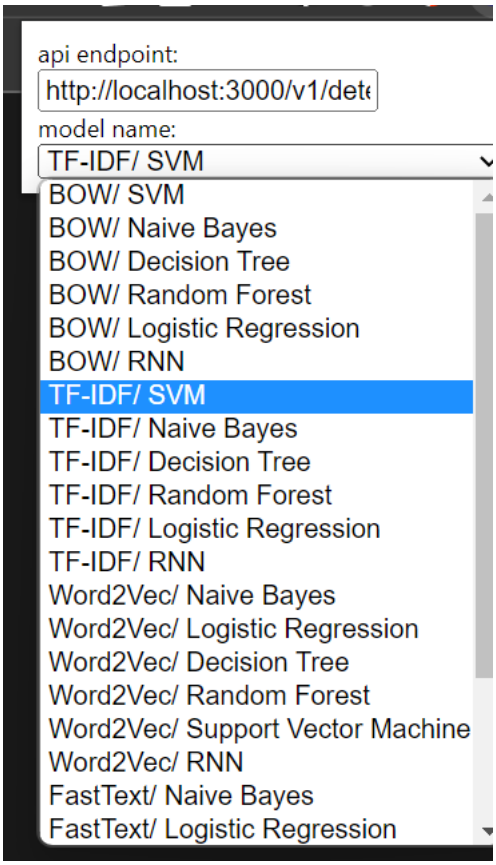


Figure 48 Chrome Plugin model selection

5.3. Cloud Deployment

The model and system is tried to perform into the cloud. The cloud is choosing AWS (Amazon Web Services) to deploy the system.

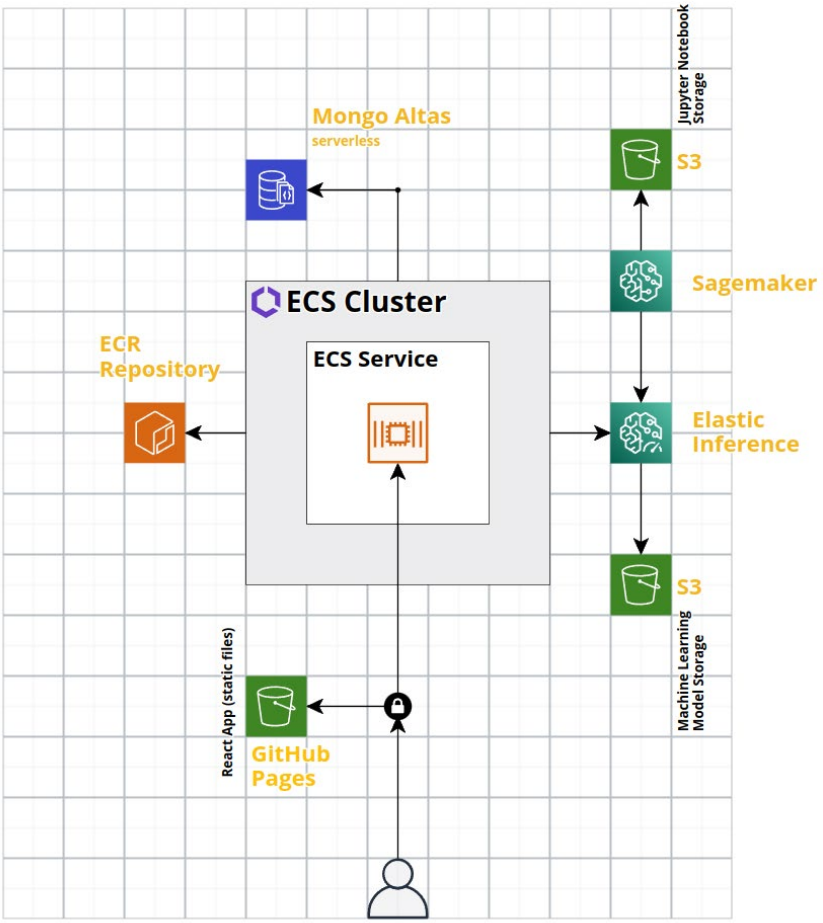


Figure 49 AWS Cloud Infrastructure

The system is deployed into cloud. First, the frontend will be using GitHub Pages, since the react is the client side rendering application. Therefore, it will be host on the GitHub Pages. It will be a free services for hosting the website. However, in order to make the services successful, backend api support is needed.

The business logic is implement in the express api server. It will be containerization and register into the AWS ECR services, it is a services that allow developer to register the docker image and pull to other devices. After that, the express api server image will be

host on ECS cluster, which opened a EC2 instance for hosting the server.

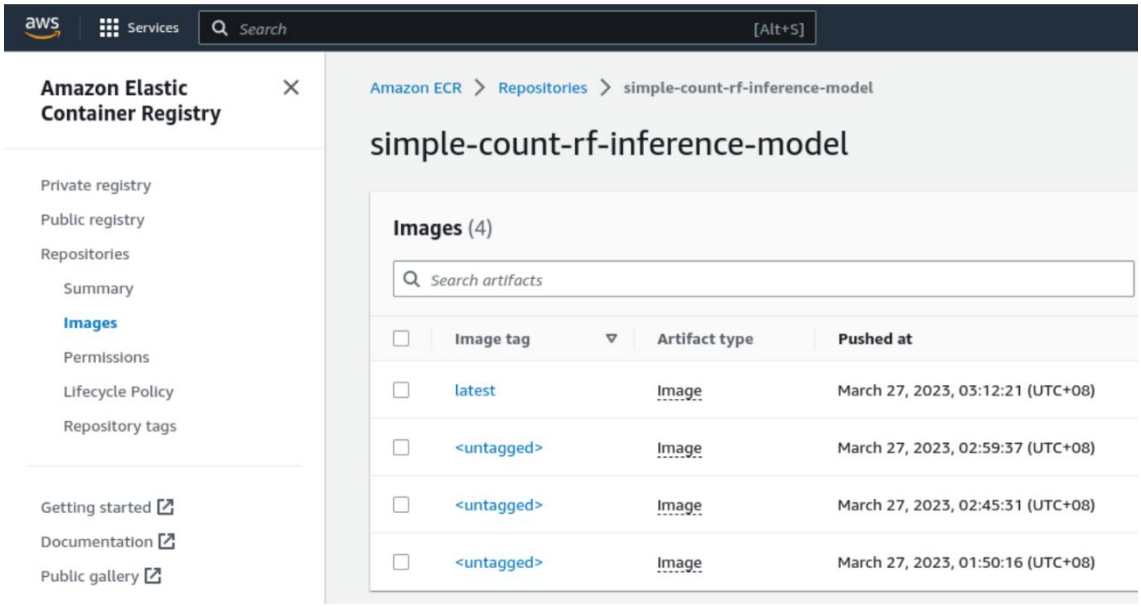


Figure 50 Amazon ECR

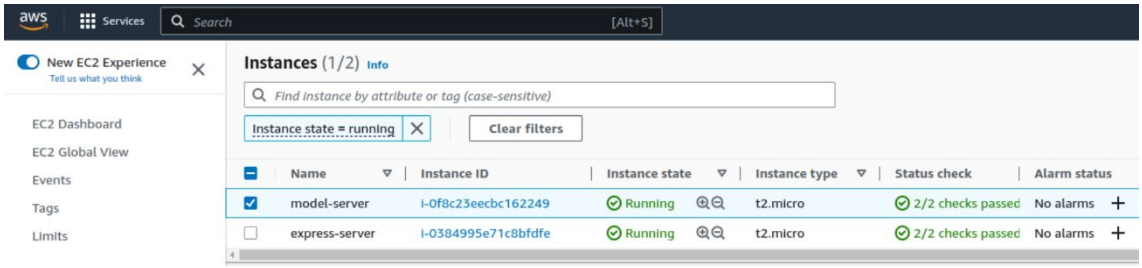


Figure 51 Amazon EC2 Instance

On the other hand, the model have a model server to perform the detection process. However, in deployment, AWS SageMaker is used for hosting the model service. The model server will be a back-up plan and used for the case running in standalone service.

The AWS SageMaker is a machine learning service provided by AWS, it allow to upload the custom model to AWS S3 Bucket Cloud Storage, and use a custom container

to package the processing step of the model, including the text-preprocessing and value post-processing, like tokenization and softmax process.

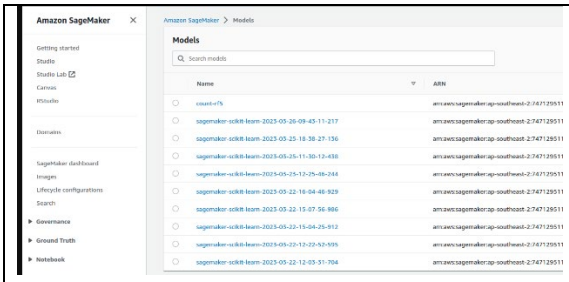


Figure 52 AWS SageMaker model

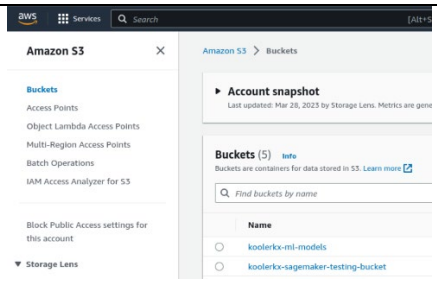


Figure 53 AWS S3 Bucket to store the model

In additional, the express server depends on a mongodb to store the dataset and user detected history. MongoDB Atlas is chosen as the cloud services.'

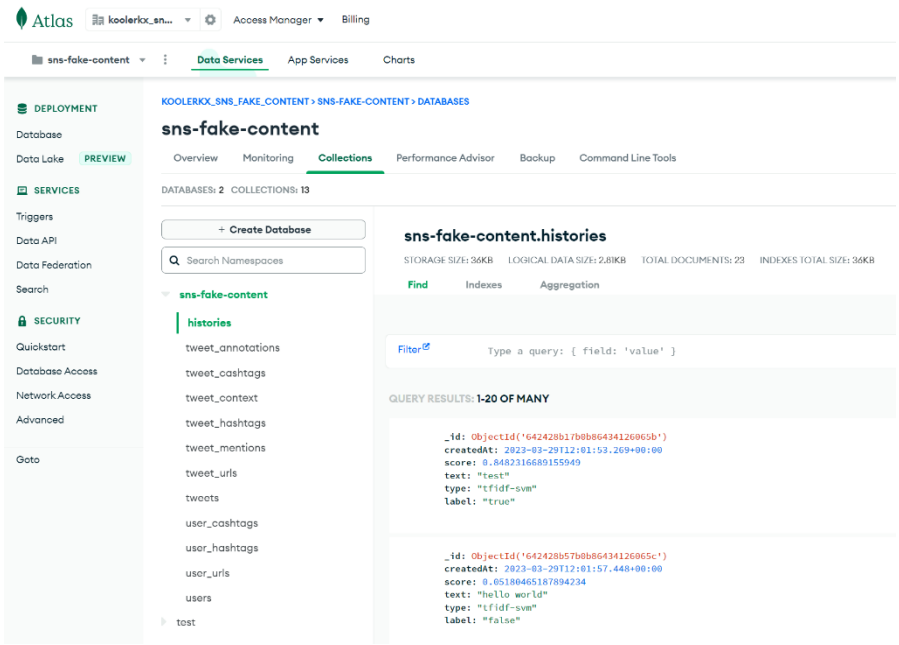


Figure 54 MongoDB Atlssas

5.4. Model Result

This study aimed to compare the performance of machine learning algorithm, recurrent neural network, and transformer models trained with different word embedding methods, such as Word2Vec and fast text. The comparison was made using various metrics, including the PR curve, ROC curve AUC score, accuracy, precision score, recall score, and F1 score.

The dataset is slice into different size for increasing the speed of model training and testing purpose. It will take the amount of data for each class. (The 3xs data will have 6000 record, 3000 for true and false each class).

ss

Size	Amount
2xs	6000
xs	20000
s	160000
m	320000
l	~1000000
Xl	All data (~1300000)

According to the test plan in the methodology, the model is trained to compare its performance.

label	NB accuracy	LR accuracy	DT accuracy	RF accuracy	SVM accuracy	RNN accuracy
str	f64	f64	f64	f64	f64	f64
"BOW"	0.83925	0.864	0.80875	0.8625	0.8685	null
"TF-IDF"	0.83925	0.86525	0.79875	0.85325	0.886	null
"FastText"	0.5555	0.5755	0.621	0.68075	0.60975	0.884219
"Word2Vec"	0.55525	0.575	0.60975	0.68375	0.58925	0.882156

Figure 55 Testing score result.

By applying different method to train the model. The result of the model is collected for evaluation. From the result across all method and word embedding method (except transformer model), the Word2Vec and FastText word embedding with recurrent neural network (RNN) is the best for general model. Detail will be explained below.

Different machine learning algorithm is applied, include Naive Bayes (NB), Logistic Regression (LR), Decision Tree (DT), Random Forest (RF), Support Vector Machine (SVM) and Recurrent Neural Network (RNN).

The following comparison will focus on accuracy, other metrics may also used as a supplementary information.

5.4.1. Bag of Words

	label	nb_pipeline	lr_pipeline	dt_pipeline	rf_pipeline	svm_pipeline
auc	BOW	0.917222	0.927524	0.820194	0.936146	0.942469
accuracy	BOW	0.839250	0.864000	0.808750	0.862500	0.868500
precision	BOW	0.865695	0.881726	0.844707	0.896296	0.879741
recall	BOW	0.812012	0.848145	0.767578	0.827148	0.860840
f1	BOW	0.837994	0.864609	0.804298	0.860335	0.870188

Figure 56 BOW model result

First, the simplest way of word embedding is a bags of word model (BOW). From the bag of word model, all the models achieve more than 80% accuracy. The support vector machine perform the best in the BOW model. The logistic regression and random forest also close to the best. The decision tree performs the worst. This may because the decision tree may take too few features to determine the result. Naive Bayes perform in middle range which show probability may not the best model for BOW model classification.

Here is some training result with the confusion matrix:

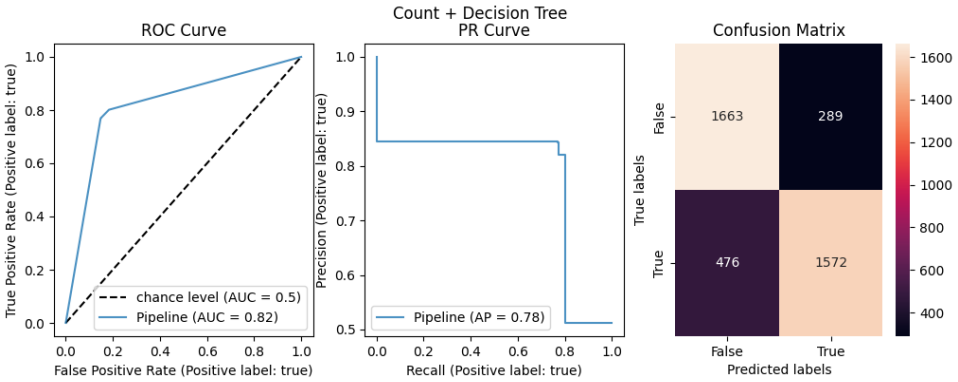


Figure 57 BOW + Decision Tree

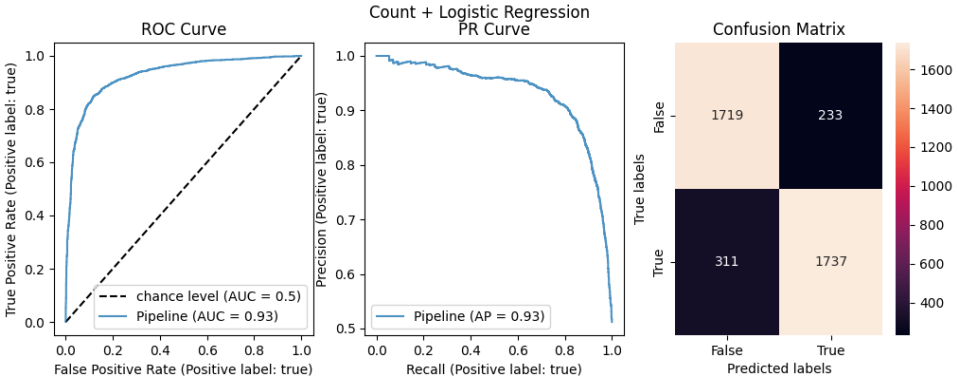


Figure 58 BOW + Logistic Regression

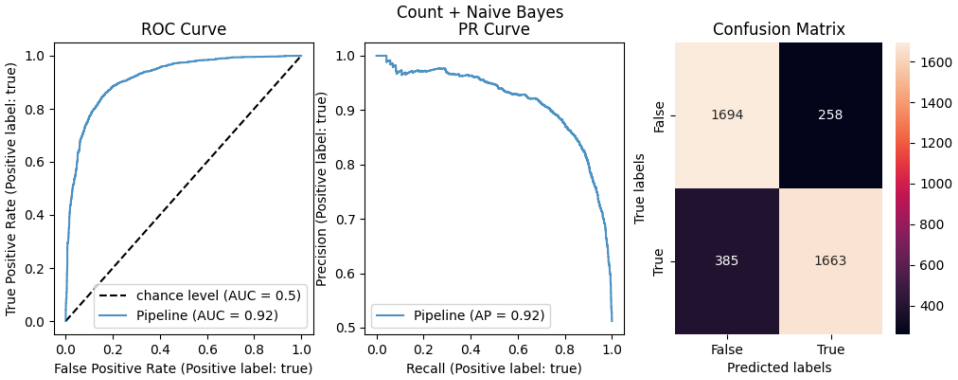


Figure 59 BOW + Naive Bayesss

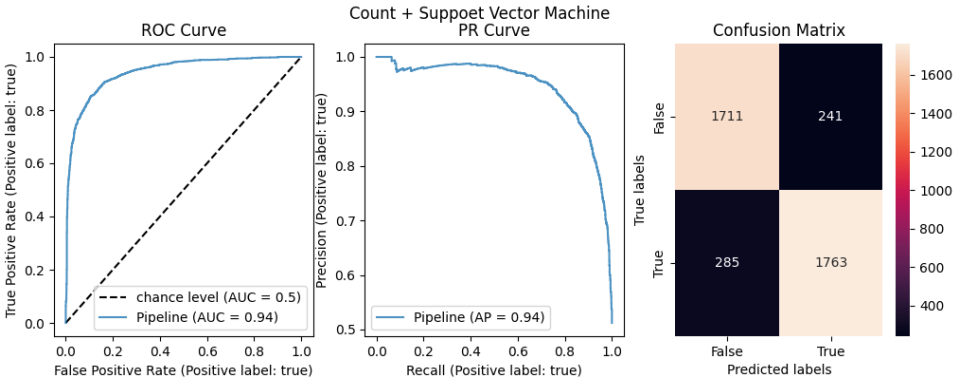


Figure 60 BOW + SVM

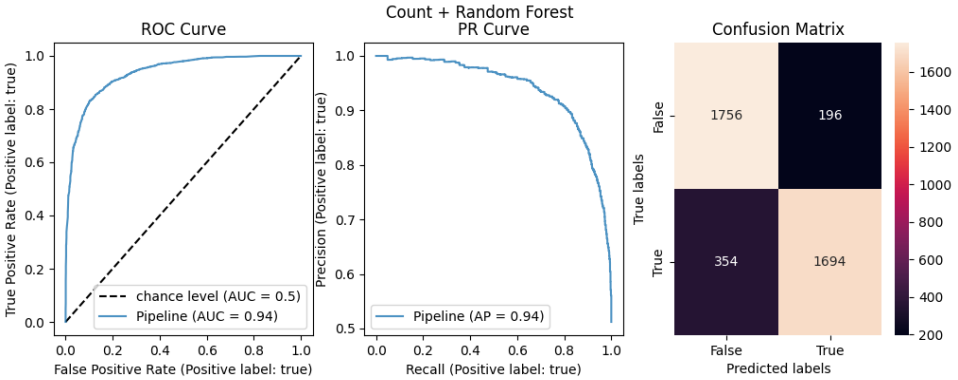


Figure 61 BOW + Random Forest

5.4.2. TF-IDF

	label	nb_pipeline	lr_pipeline	dt_pipeline	rf_pipeline	svm_pipeline
auc	TF-IDF	0.917222	0.934932	0.806762	0.933560	0.954285
accuracy	TF-IDF	0.839250	0.865250	0.798750	0.853250	0.886000
precision	TF-IDF	0.865695	0.876309	0.825905	0.877519	0.893670
recall	TF-IDF	0.812012	0.857910	0.769043	0.829102	0.882324
f1	TF-IDF	0.837994	0.867012	0.796460	0.852624	0.887961

Figure 62 TF-IDF Model Result

Although the data are being processed before training, the stopword is already removed.

However, some word may still exist may times. Therefore, TF-IDF is used to do another word embedding method which aimed to reduce the impact of frequent word.

The result of TF-IDF embedding show that support vector machine, random forest and logistic regression is the most suitable model for TF-IDF embedding, which result in 88% and 86% accuracy with respectively. Same as BOW model, decision tree perform the worst and naive bayes in the middle range.

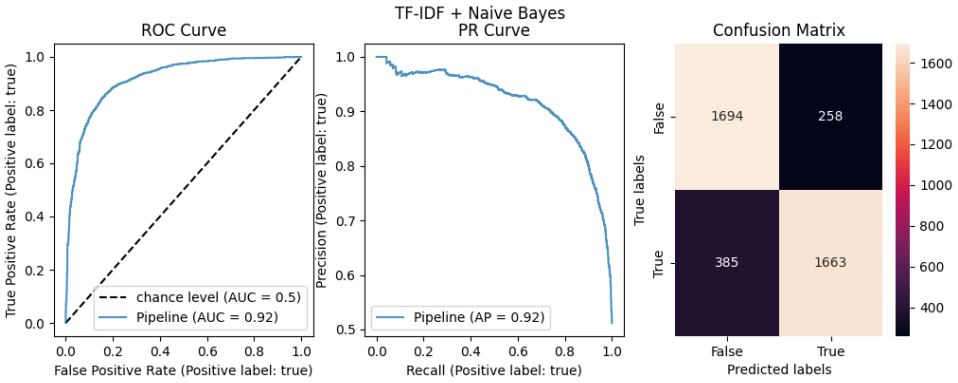


Figure 63 TF-IDF + Naive Bayes

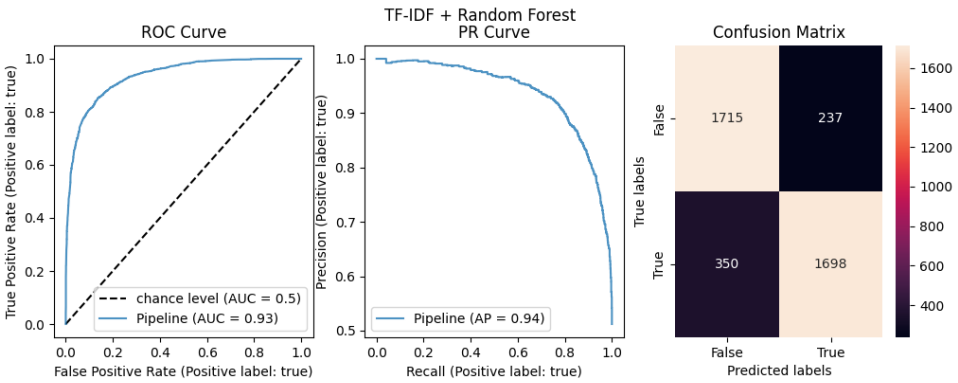


Figure 64 TF-IDF Radom Forest

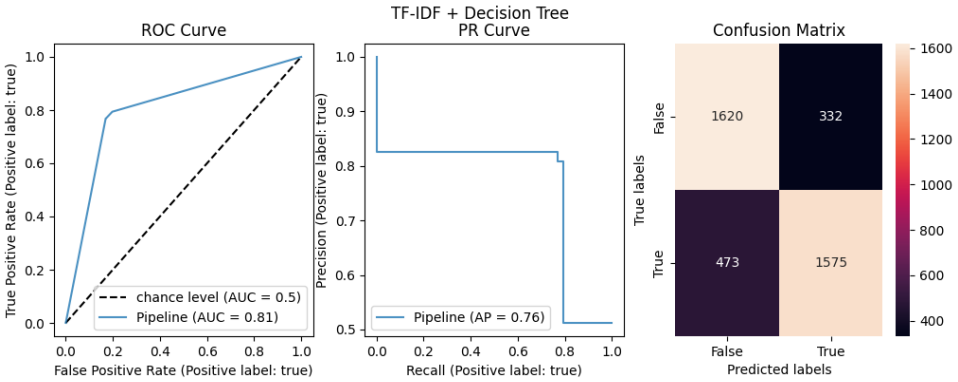


Figure 65 TF-IDF + Decision Tree

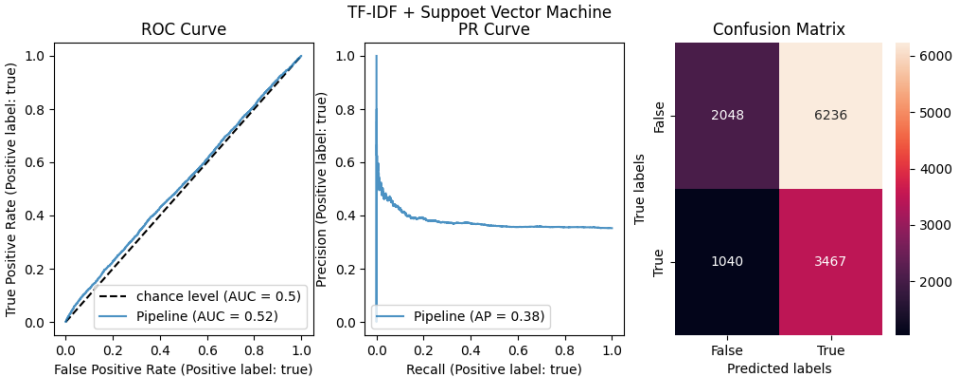


Figure 66 TF-IDF + SVM

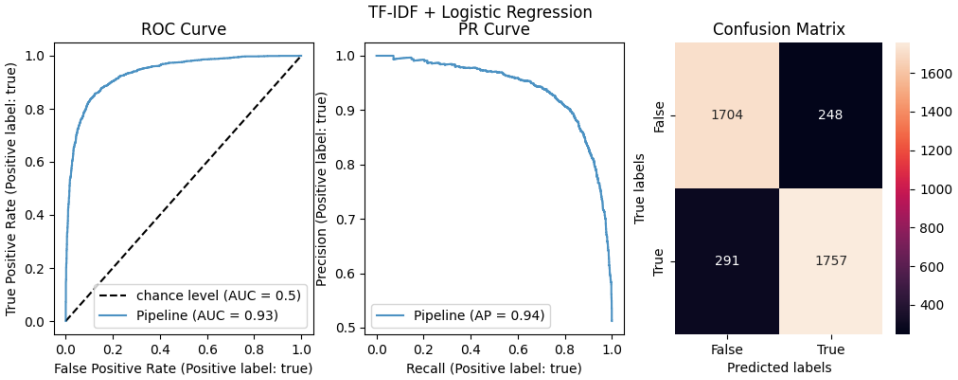


Figure 67 TF-IDF + Logistic Regression

5.4.3. Word2Vec

Moving from from the term frequency model to similarity model. The Word2Vec model trained by the current dataset FakeNewsNet.

	section	correct	incorrect	score
0	capital-common-countries	114	392	0.225296
1	capital-world	128	1035	0.110060
2	currency	31	77	0.287037
3	city-in-state	270	1866	0.126404
4	family	376	44	0.895238
5	gram1-adjective-to-adverb	273	483	0.361111
6	gram2-opposite	275	277	0.498188
7	gram3-comparative	756	56	0.931034
8	gram4-superlative	414	48	0.896104
9	gram5-present-participle	338	82	0.804762
10	gram6-nationality-adjective	213	882	0.194521
11	gram7-past-tense	559	197	0.739418
12	gram8-plural	620	82	0.883191
13	gram9-plural-verbs	225	81	0.735294
14	Total accuracy	4592	5602	0.450461

Figure 68 Word2Vec Benchmark Score

By using a Gensim library provided benchmark similarity evaluation, it's total accuracy result in 0.2. Although the score is low, this does not mean the model is bad, since it have to depends on the dataset and the scenario to evaluate. The benchmark is for general purpose.

	breaking	breaking_cos	apple	apple_cos	human	human_cos	vote	vote_cos	vehicle	vehicle_cos	automobile	automobile_cos	accident	accident_cos
0	breakingbad	0.929433	applebee	0.847189	subhuman	0.874830	votevotevote	0.894970	vehicles	0.751481	mobile	0.758848	caraccident	0.889498
1	breakingdawn	0.889907	applecup	0.819756	humanly	0.861708	ivote	0.872577	cubicle	0.590498	automation	0.618116	accidental	0.857783
2	breakin	0.859077	appl	0.783729	inhuman	0.837466	votem	0.840758	car	0.470535	automotive	0.615731	accidentally	0.786618
3	breaki	0.826572	grapple	0.759658	humanoid	0.816815	govote	0.838701	cle	0.457736	mobil	0.611205	accio	0.682830
4	breakingnew	0.781423	pineapple	0.730572	humankind	0.784458	votedem	0.827706	popsicle	0.420781	mobilephone	0.610657	incident	0.640525
5	breakingnews	0.718127	pple	0.725114	humanright	0.772316	votegop	0.824126	motorcycle	0.411498	automate	0.605889	trident	0.610390
6	groundbreaking	0.704342	applehead	0.707455	humans	0.756182	voteoutgop	0.799495	cycle	0.410812	mobilemarkete	0.565845	président	0.557403
7	leaking	0.695665	applenews	0.689499	humane	0.752255	voten	0.794324	tentacle	0.410409	mobilise	0.561255	accsc	0.536898
8	braking	0.671086	applegate	0.642830	humayun	0.749160	covote	0.792506	bicycle	0.392035	exxonmobil	0.533392	flaccid	0.522833
9	freaking	0.660106	appleton	0.631131	humanize	0.744237	votegopout	0.786066	barnacle	0.384174	automaker	0.530886	acc	0.517022

Figure 69 Word2Vec Similarity Word

When listing some word with the top similar word with it, some word are perform good in human sense. For example, vehicle is related to car, cycle, motorcycle. The accident related to caraccident.

	label	nb_pipeline	lr_pipeline	dt_pipeline	rf_pipeline	svm_pipeline	RNN
auc	Word2Vec	0.587313	0.590743	0.609990	0.765740	0.627961	0.946212
accuracy	Word2Vec	0.555250	0.575000	0.609750	0.683750	0.589250	0.882156
precision	Word2Vec	0.552519	0.583493	0.623667	0.693524	0.578824	0.882646
recall	Word2Vec	0.690918	0.593750	0.599609	0.685059	0.726074	0.882261
f1	Word2Vec	0.614016	0.588577	0.611402	0.689266	0.644141	0.882454

Figure 70 Word2Vec Model Result

The traditional machine learning algorithm perform not as well as term frequency model. This may because the word2vec model have too many feature which produce a large matrix. This may lead to overfit into the training dataset. Therefore, it perform not so well in the testing dataset.

Since the traditional algorithm is not working well on the similarity model. The neural work is tried to applied. By using the recurrent neural network, using gate recurrent unit (GRU) to handle the context across the sentence.

```
Model: "Word2Vec_014113"
_________________________________________________________________
 Layer (type)                 Output Shape              Param #
=================================================================
 embedding_1 (Embedding)      (None, None, 300)         12709500

 gru_4 (GRU)                  (None, None, 256)         428544

 gru_5 (GRU)                  (None, None, 128)         148224

 gru_6 (GRU)                  (None, None, 64)          37248

 gru_7 (GRU)                  (None, 32)                9408

 dense_3 (Dense)              (None, 32)                1056

 dense_4 (Dense)              (None, 16)                528

 dense_5 (Dense)              (None, 1)                 17

=================================================================
Total params: 13,334,525
Trainable params: 625,025
Non-trainable params: 12,709,500
_________________________________________________________________
```

Figure 71 Word2Vec RNN

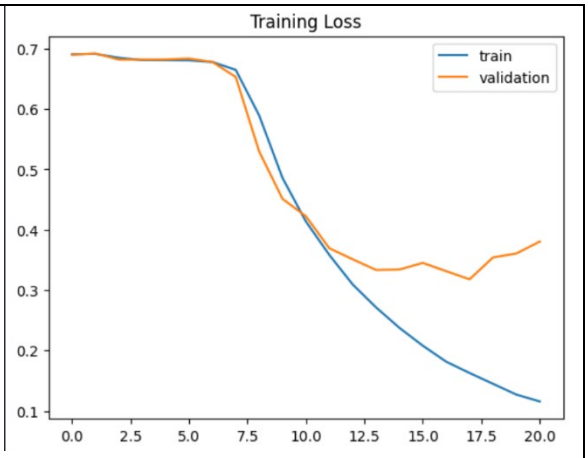


Figure 72 Word2Vec RNN training loss

The model is built by 4 GRU layer, there is a dropout layer between the GRU layer to prevent to overfit saturation. This result 88% accuracy. This show that the neural network is suitable for the similarity word embedding method to do classification.

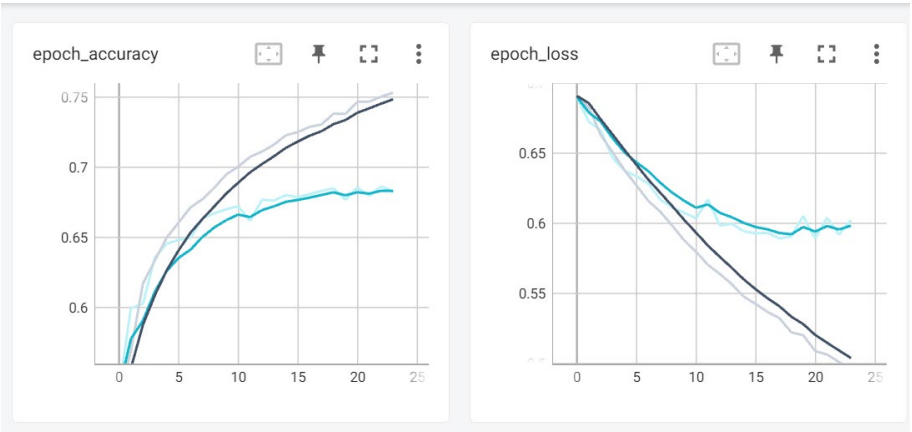


Figure 73 Word2Vec Training Result

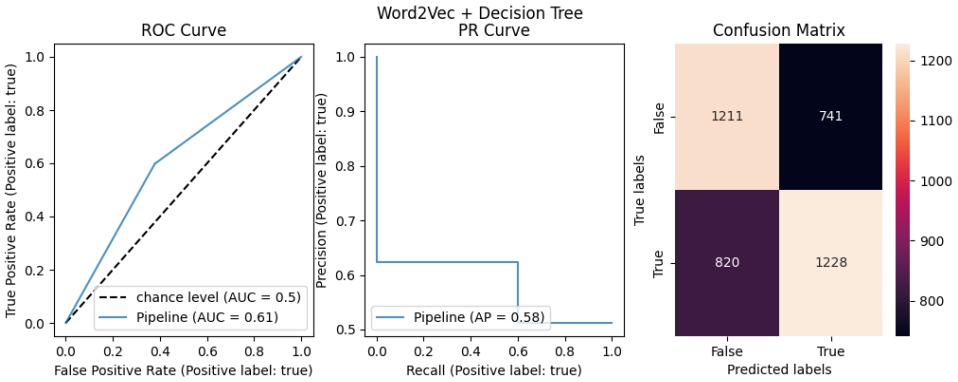


Figure 74 Word2Vec + Decision Tree

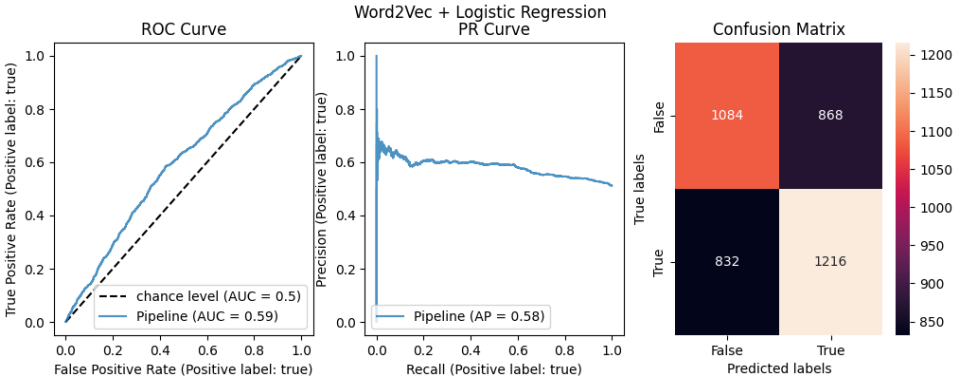


Figure 75 Word2Vec + Logistic Regression

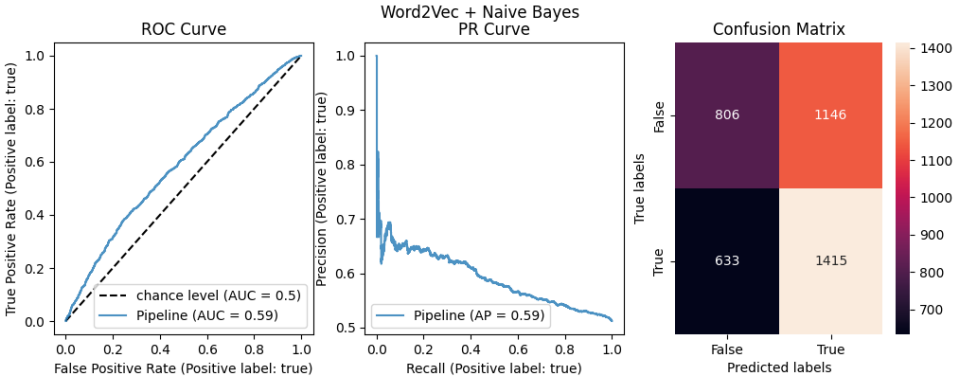


Figure 76 Word2Vec + Naive Bayes

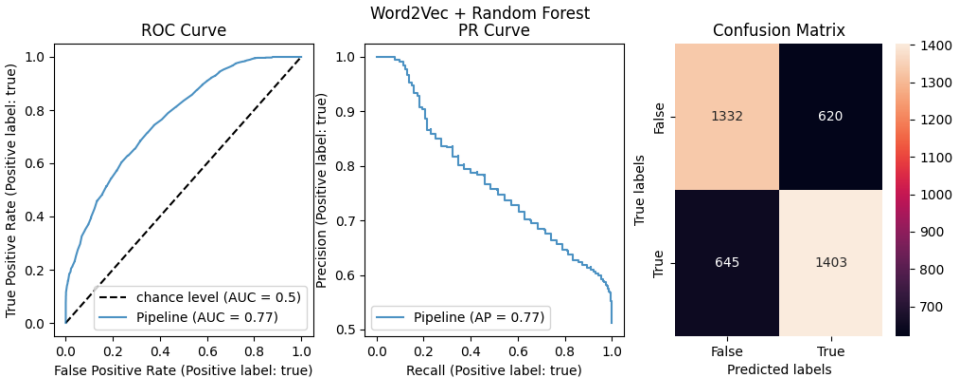


Figure 77 Word2Vec + Random Forest

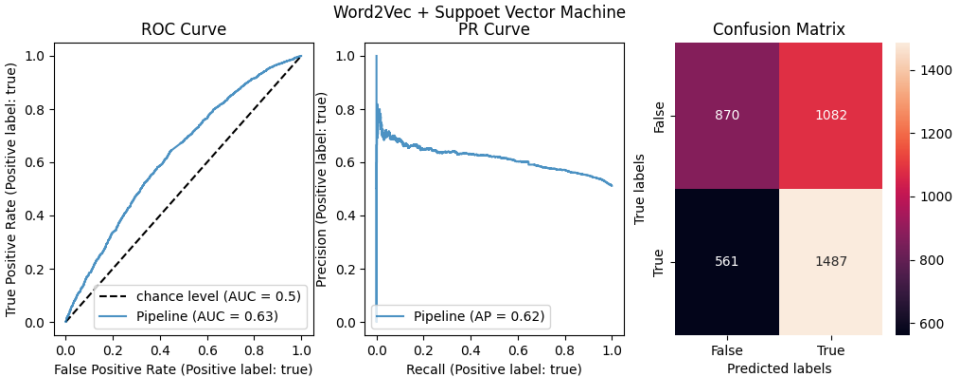


Figure 78 Word2Vec + SVM

5.4.4. FastText

Apart from the word2vec model, another similarity model also tried to compare. FastText which developed by Facebook, are another similarity model for word embedding.

	section	correct	incorrect	score
0	capital-common-countries	5	501	0.009881
1	capital-world	3	1160	0.002580
2	currency	0	108	0.000000
3	city-in-state	9	2127	0.004213
4	family	38	382	0.090476
5	gram1-adjective-to-adverb	488	268	0.645503
6	gram2-opposite	397	155	0.719203
7	gram3-comparative	175	637	0.215517
8	gram4-superlative	175	287	0.378788
9	gram5-present-participle	46	374	0.109524
10	gram6-nationality-adjective	441	654	0.402740
11	gram7-past-tense	14	742	0.018519
12	gram8-plural	377	325	0.537037
13	gram9-plural-verbs	136	170	0.444444
14	Total accuracy	2304	7890	0.226015

Figure 79 FastText Model Benchmark Score

From the benchmark of the similarity model. The score of the fast text model is less than the word2vec model. It's total accuracy score is 0.2. Again, the similarity model benchmark not suitable on this use case. So the low score of the similarity is not necessary a concern of building the machine learning model.

	label	nb_pipeline	lr_pipeline	dt_pipeline	rf_pipeline	svm_pipeline	RNN
auc	FastText	0.578161	0.591565	0.620498	0.764190	0.642147	0.950726
accuracy	FastText	0.555500	0.575500	0.621000	0.680750	0.609750	0.884219
precision	FastText	0.549889	0.584135	0.631164	0.685426	0.595603	0.874567
recall	FastText	0.726562	0.593262	0.625000	0.695801	0.740723	0.897843
f1	FastText	0.625999	0.588663	0.628067	0.690574	0.660283	0.886053

Figure 80 FastText Model Result

The result of model using fast text as word embedding is similar to the result of word2vec, only RNN having a high accuracy. The random forest also having the highest score across different machine learning algorithm, which is same as word2vec method's model.

```
Model: "Fasttext_RNN_010233"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 embedding (Embedding)       (None, None, 500)         21182500

 gru (GRU)                   (None, None, 256)         582144

 gru_1 (GRU)                 (None, None, 128)         148224

 gru_2 (GRU)                 (None, None, 64)          37248

 gru_3 (GRU)                 (None, 32)                9408

 dense (Dense)               (None, 32)                1056

 dense_1 (Dense)             (None, 16)                528

 dense_2 (Dense)             (None, 1)                 17

=================================================================
Total params: 21,961,125
Trainable params: 778,625
Non-trainable params: 21,182,500
_________________________________________________________________
```

Figure 81 FastText RNN

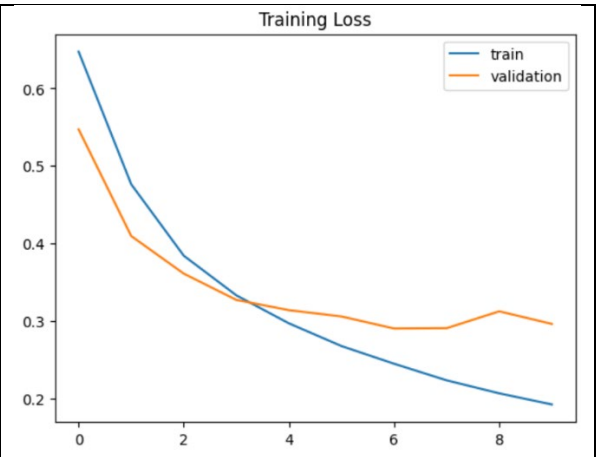


Figure 82 FastText RNN Training Score

The structure of the neural network trained with FastText embedding method is same as the word2vec. However, the epoch trained is slightly lower than word2vec model. It is applying the early stopping in 9 epochs.

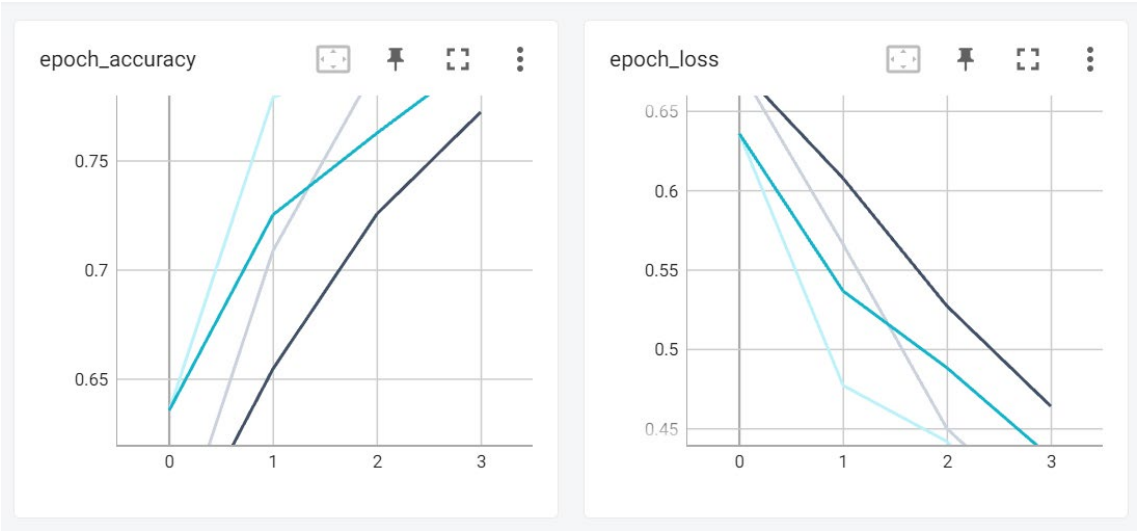


Figure 83 FastText Training Result

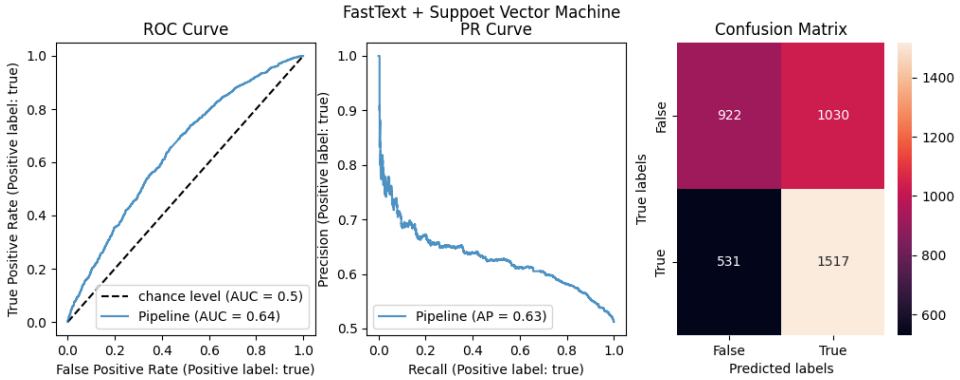


Figure 84 FastText + SVM

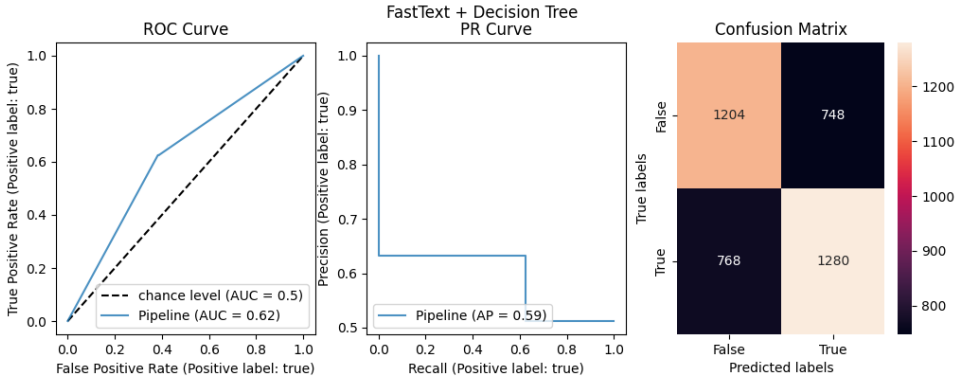


Figure 85 FastText + Decision Tree

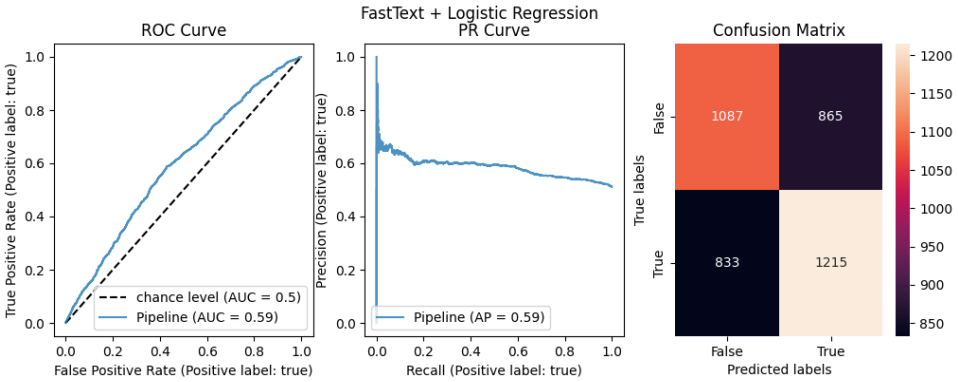


Figure 86 FastText + Logistic Regression

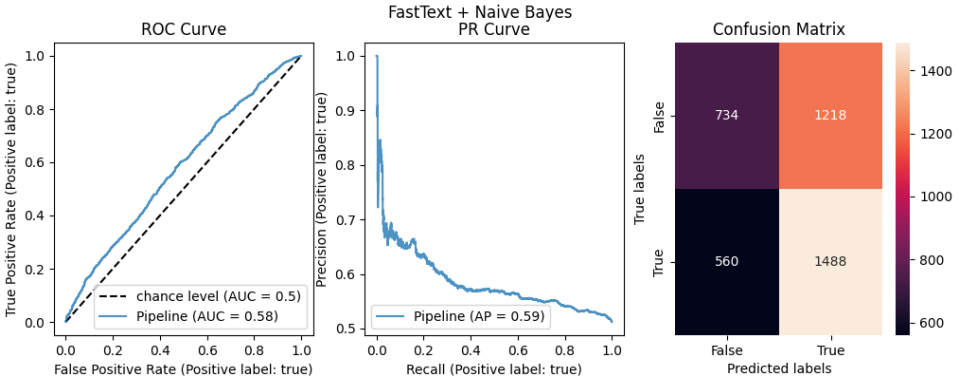


Figure 87 FastText + Naive Bayes

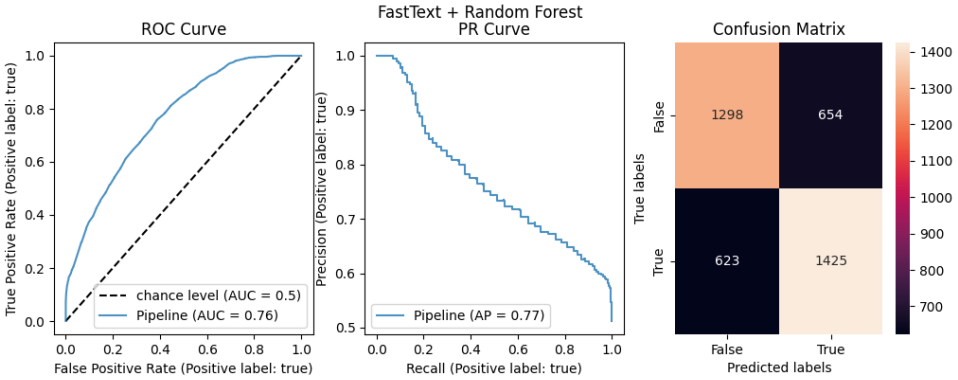


Figure 88 FastText + Naive Bayes

5.4.5. Transformer

The transformer model is a deep learning approach to do the classification. It has a special tokenization stessp which will insert some special token into the text. For example, the padding will insert [PAD] into the text string like the following.

```
i've been waiting for a huggingface course my whole life. [PAD] [PAD] [PAD] [PAD] [PAD] [PAD] [PA
```

The transformer model has many branch, as mentioned, BERT and XLNet was chosen for solving this task. To fine tune the parameter. Different token size is tried. Also, it is applied different size of the data to see whether the final result will affect by the amount of data. Other hyper parameter is fixed so that could compare the token size and data size only.ss

	Size	Token Size	Accuracy
BERT	2xs	128	0.83
	xs	128	0.92
	s	128	0.96
	m	128	0.97
	l	128	0.98
	xl	128	0.97
	m	32	0.92
	m	16	0.95
XLNet	m	32	0.94

The result table is extracted the best checkpoint in the training process. So that it could minimize the risk of overfitting the validation data. From above result table. It can be observed that all the accuracy are above 90%, except the 2xs data size (taking 3000 sample from each class). The highest accuracy is using large (l) dataset which result in 98% accuracy. Another model XLNet trained in 32 token size, which is a related small value, result in 94% accuracy.

Although all model is result in more than 90% accuracy. This make a question that is it overfitting the dataset. However, consider with the complexity of the problem. The binary classification is not so hard as multi-class classification. Therefore, it may possible to achieving a very high accuracy.

Training Result:

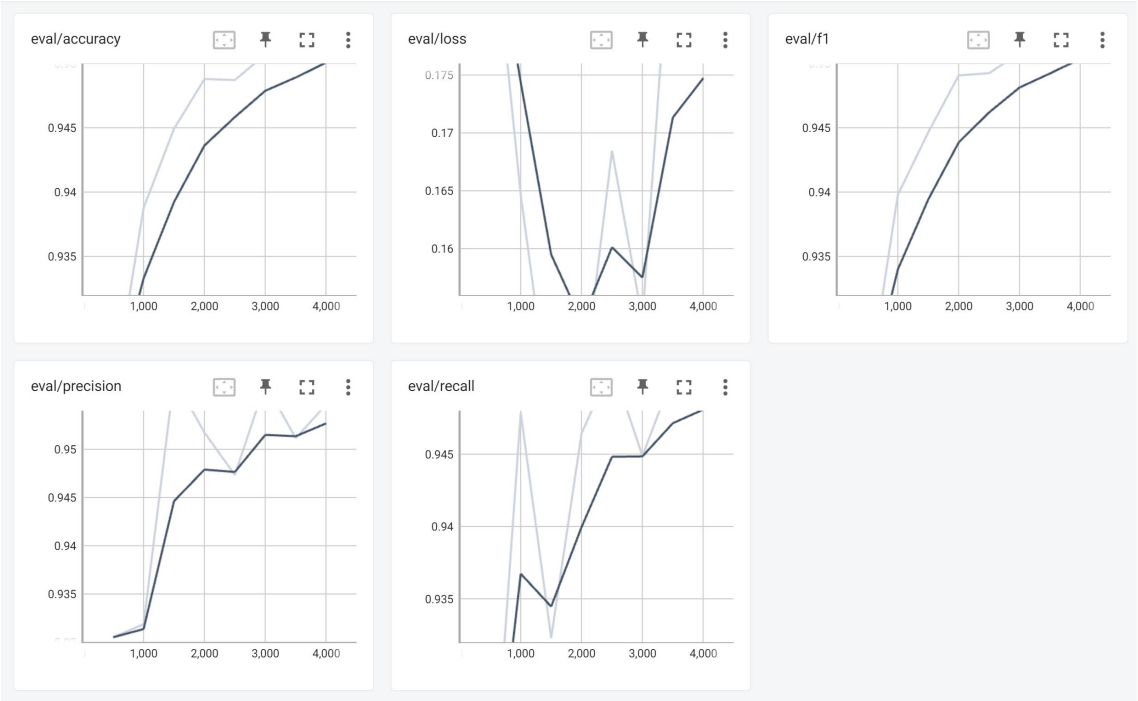


Figure 89 XLNet training result

6. Project Timeline

The project is divided to 3 phase, which is the “research and data preprocessing”, “analysis and modeling” and implementation. Before the project start, literary research will also be doing before entering the main development stage of the project. Here is an overview of the timeline:

Phase	Tasks/ Goals	Date
Phase 0	Literary Research	Sept and Oct
Phase 1	Data Processing/ ETL	Oct
	Data Explore	Oct and Nov
Phase 2	Detection Model Build and Train	Nov - Jan
	Model Comparison	Dec - Jan
	Text Analysis/ Text Mining	Dec - Feb
	Construct Analysis Dashboard and Testing	Jan - Feb
	System Design for AI Application	Jan - Feb
Phase 3	Detection Model Implementation	Feb
	Refined Application design/UI	March
	Final Report and Buffer Time	March

Table 5 Project timeline

6.1. Project Stage

6.1.1. Phase 1: Research and Data Preprocessing

Phase 1 will focus on the data processing, which prepare the data for building the machine learning model and some simple exploration will be done on this stage.

This phase should be end before November, so that next phase would have enough content to start the development.

6.1.2. Phase 2: Analysis and Modeling

Phase 2 will be the main development stage. The data will be used to do the analysis by using NLP technique and start visualizing in dashboard. Besides, the machine learning model will be training and compare in this stage. About the application, the application system design will be start in this stage.

This phase will be start after the data is prepared, it is expected to start from November 2022 and end before February 2023.

6.1.3. Phase 3: Model Implementation and Testing

Phase 3 will be implementing the model into application designed in phase 2. Before finalizing the outcome of the project, it is planned to refine all the design of this project. Including the UI design of application and dashboard, dashboard design, story presentation of the dashboard and other related item. All end-user view will be reviewed in this phase. It also going to prepare the final report and presentation.

In addition, testing will also conduct during the model implementation. Since the development will be used test-driven development. The application will be test just after development.

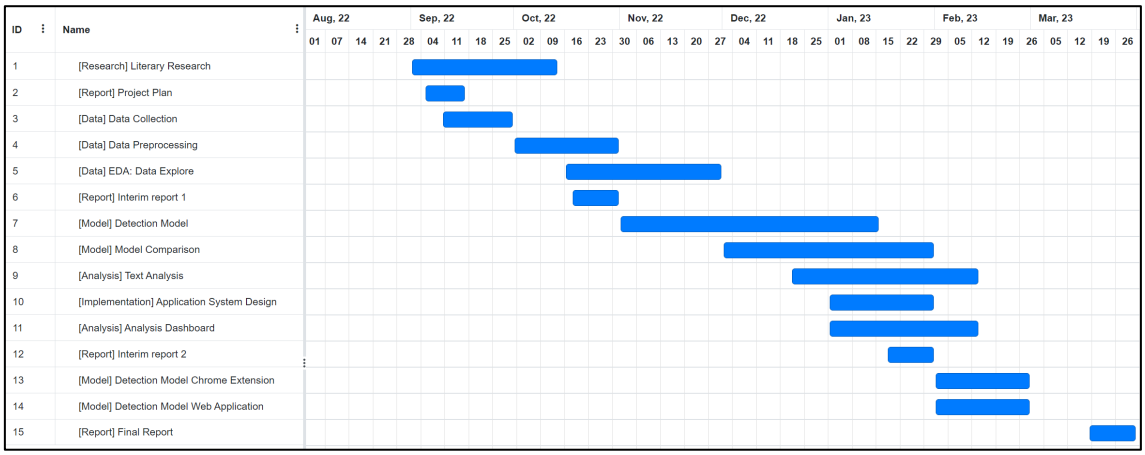


Figure 90 Project timeline - Gantt chart

7. Conclusion

In this project. it is going to perform fake content detection. It is discussed different embedding mode including BOW, TF-IDF, Word2Vec, FastText with different algorithm including Random Forest, SVM, Decision Tree, Logistic Regression and Naive Bayes, RNN GRU and transformer BERT and XLNet. It is found that SVM and Random Forest is performed well across different term frequency text embedding method. However, for the similarity embedding model, RNN is significantly better than other method. However, compare with transformer model, transformer model is the best model to do the classification task which achieve 97% accuracy.

On the other hand, the implementation method also considered, it is implemented in a web dashboard for end-user to detect the given text and perform simple analysis on the dataset. A chrome plugin also developed for end-user easily to perform the detection task. The whole system supported by backend API express server and a model server. It is deployed on cloud by make good use of AWS services and hosted the database on MongoDB atlas.

8. Reference

Ahmad, A. R., & Murad, H. R. (2020). The Impact of Social Media on Panic During the COVID-19 Pandemic in Iraqi Kurdistan: Online Questionnaire Study. *Journal of Medical Internet Research*, *22*(5), e19556. https://doi.org/10.2196/19556

Atoum, I., Otoom, A., & Kulathuramaiyer, N. (2016). A Comprehensive Comparative Study of Word and Sentence Similarity Measures. *International Journal of Computer Applications*, *135*(1), 10–17. https://doi.org/10.5120/ijca2016908259

Buntain, C., & Golbeck, J. (2017). Automatically Identifying Fake News in Popular Twitter Threads. *2017 IEEE International Conference on Smart Cloud (SmartCloud)*. https://doi.org/10.1109/smartcloud.2017.40

Carter, M., Tsikerdekis, M., & Zeadally, S. (2021). Approaches for Fake Content Detection: Strengths and Weaknesses to Adversarial Attacks. *IEEE Internet Computing*, *25*(2), 73–83. https://doi.org/10.1109/mic.2020.3032323

Chen, J., Hu, Y., Liu, J., Xiao, Y., & Jiang, H. (2019). Deep Short Text Classification with Knowledge Powered Attention. *Proceedings of the AAAI Conference on Artificial Intelligence*, *33*, 6252–6259. https://doi.org/10.1609/aaai.v33i01.33016252

Daniel, N. (2020, August 23). *What is a Confusion Matrix?* Unite.AI. https://www.unite.ai/what-is-a-confusion-matrix/

Education, I. C. (2021, August 17). *Natural Language Processing (NLP)*. https://www.ibm.com/cloud/learn/natural-language-processing

EUROPEAN COURT OF AUDITORS. (2020). EU action plan against disinformation. In *EUROPEAN COURT OF AUDITORS*. Retrieved October 23, 2022, from https://www.eca.europa.eu/lists/ecadocuments/ap20_04/ap_disinformation_en.pdf

Express - Node.js web application framework. (n.d.). https://expressjs.com/

Fake News Detection Datasets - University of Victoria. (n.d.). https://www.uvic.ca/ecs/ece/isot/datasets/fake-news/index.php

G. Purna Chandar Rao, & V. B. Narasimha. (2021). Feature-Based Learning Model for Fake News Detection and Classification. *International Journal of Scientific Research in Science and Technology*, 130–139. https://doi.org/10.32628/ijsrst2184111

Gensim: topic modelling for humans. (n.d.). https://radimrehurek.com/gensim/

Giulio Angiani, Laura Ferrari, Tomaso Fontanini, Paolo Fornacciari, Eleonora Iotti, Federico Magliani, & Stefano Manicardi. (2016). A Comparison between Preprocessing Techniques for Sentiment Analysis in Twitter. *KDWeb*. http://ceur-ws.org/Vol-1748/paper-06.pdf

Hall, K., Chang, V., & Jayne, C. (2022). A review on Natural Language Processing Models for COVID-19 research. *Healthcare Analytics*, *2*, 100078. https://doi.org/10.1016/j.health.2022.100078

Haralambous, Y., Kim-Dufor, D. H., Lenca, P., Billot, R., Ryan, T. C., Marsh, J., DeVylder, J., Walter, M., Berrouiguet, S., & Lemey, C. (2019). Machine Learning and Natural Language Processing in Mental Health: Systematic Review (Preprint). *Journal of Medical Internet Research*. https://doi.org/10.2196/preprints.15708

Jingbo, Z., & Tianshun, Y. (2002). A knowledge-based approach to text classification. *Proceeding of the First SIGHAN Workshop on Chinese Language Processing* -. https://doi.org/10.3115/1118824.1118844

Kadhim, A. I. (2018). An Evaluation of Preprocessing Techniques for Text Classification. *International Journal of Computer Science and Information Security*, *16*(6), 22–32.

Khanam, Z., Alwasel, B. N., Sirafi, H., & Rashid, M. (2021). Fake News Detection Using Machine Learning Approaches. *IOP Conference Series: Materials Science and Engineering*, *1099*(1), 012040. https://doi.org/10.1088/1757-899x/1099/1/012040

Koloski, B., Stepišnik Perdih, T., Robnik-Šikonja, M., Pollak, S., & Škrlj, B. (2022). Knowledge graph informed fake news classification via heterogeneous representation ensembles. *Neurocomputing*, *496*, 208–226. https://doi.org/10.1016/j.neucom.2022.01.096

Language Processing Pipelines. (n.d.). spaCy. https://spacy.io/usage/processing-pipelines

Mohtaj, S., & Möller, S. (2022). The Impact of Pre-processing on the Performance of Automated Fake News Detection. *Lecture Notes in Computer Science*, 93–102. https://doi.org/10.1007/978-3-031-13643-6_7

MongoDB. (n.d.). *MongoDB: The Developer Data Platform*. https://www.mongodb.com/

Nadkarni, P. M., Ohno-Machado, L., & Chapman, W. W. (2011). Natural language processing: an introduction. *Journal of the American Medical Informatics Association*, *18*(5), 544–551. https://doi.org/10.1136/amiajnl-2011-000464

Node.js. (n.d.). Node.js. https://nodejs.org/en/

Pierri, F., & Ceri, S. (2019a). False News On Social Media: A Data-Driven Survey. *Social and Information Networks*. https://arxiv.org/pdf/1902.07539.pdf

Pierri, F., & Ceri, S. (2019b). False News On Social Media. *ACM SIGMOD Record*, *48*(2), 18–27. https://doi.org/10.1145/3377330.3377334

Rate limits. (n.d.). Docs | Twitter Developer Platform. https://developer.twitter.com/en/docs/twitter-api/rate-limits

React – A JavaScript library for building user interfaces. (n.d.). React. https://reactjs.org/

Sajini, G., & Kallimani, J. S. (Eds.). (2021). *A Detailed Survey Study on Classification and Various Attributes of Fake News on Social Media* (Vol. 192). Lecture Notes in Networks and Systems. https://doi.org/10.1007/978-981-33-6546-9_53

scikit-learn: machine learning in Python — scikit-learn 1.2.1 documentation. (n.d.). https://scikit-learn.org/stable/

Selva Birunda, S., & Kanniga Devi, R. (2021). A Review on Word Embedding Techniques for Text Classification. *Innovative Data Communication Technologies and Application*, 267–281. https://doi.org/10.1007/978-981-15-9651-3_23

Shu, K., Mahudeswaran, D., Wang, S., Lee, D., & Liu, H. (2020). FakeNewsNet: A Data Repository with News Content, Social Context, and Spatiotemporal Information for Studying Fake News on Social Media. *Big Data*, *8*(3), 171–188. https://doi.org/10.1089/big.2020.0062

spaCy · Industrial-strength Natural Language Processing in Python. (n.d.). https://spacy.io/

Sun, S., Luo, C., & Chen, J. (2017). A review of natural language processing techniques for opinion mining systems. *Information Fusion*, *36*, 10–25. https://doi.org/10.1016/j.inffus.2016.10.004

Text classification: a recent overview. (2005). *Annual Conference on Computers*, 125.

Thangaraj, M., & Sivakami, M. (2018). Text Classification Techniques: A Literature Review. *Interdisciplinary Journal of Information, Knowledge, and Management*, *13*, 117–135. https://doi.org/10.28945/4066

Three-Tier Architecture. (2021, August 23). https://www.ibm.com/hk-en/cloud/learn/three-tier-architecture

Transformer: A Novel Neural Network Architecture for Language Understanding. (2017, August 31). https://ai.googleblog.com/2017/08/transformer-novel-neural-network.html

typing — Support for type hints. (n.d.). Python Documentation. https://docs.python.org/3/library/typing.html

Wang, Q., Li, W., & Jin, Z. (2021). Review of Text Classification in Deep Learning. *OALib*, *08*(03), 1–8. https://doi.org/10.4236/oalib.1107175

Wang, W. Y. (2017). "Liar, Liar Pants on Fire": A New Benchmark Dataset for Fake News Detection. *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*. https://doi.org/10.18653/v1/p17-2067

Welcome to Flask — Flask Documentation (2.2.x). (n.d.). https://flask.palletsprojects.com/en/2.2.x/

Wu, H., Liu, Y., & Wang, J. (2020). Review of Text Classification Methods on Deep Learning. *Computers, Materials &Amp; Continua*, *63*(3), 1309–1321. https://doi.org/10.32604/cmc.2020.010172